

FINAL REPORT

PROJECT A-831

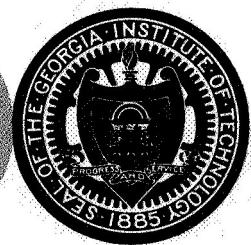
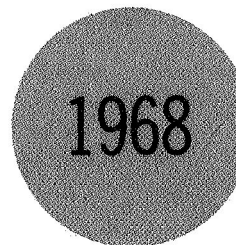
STUDY OF THE METHODS FOR THE NUMERICAL SOLUTION
OF ORDINARY DIFFERENTIAL EQUATIONS

L. J. GALLAHER, G. E. DUNCAN, O. B. FRANCIS, JR.,
J. M. GWYNN, JR., AND I. E. PERLIN

Contract NAS8-20014

26 January 1967 to 25 January 1968

Performed for
George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Huntsville, Alabama



Engineering Experiment Station
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia

N70-74566

(ACCESSION NUMBER)

205

(PAGES)

CR-102703

(NASA CR OR TMX OR AD NUMBER)

(THRU)

None

(CODE)

(CATEGORY)



GEORGIA INSTITUTE OF TECHNOLOGY
Engineering Experiment Station
Atlanta, Georgia

FINAL REPORT

PROJECT A-831

STUDY OF THE METHODS FOR THE NUMERICAL SOLUTION
OF ORDINARY DIFFERENTIAL EQUATIONS

By

L. J. GALLAHER, G. E. DUNCAN, O. B. FRANCIS, JR.,
J. M. GWYNN, JR., and I. E. PERLIN

CONTRACT NAS8-20014

26 JANUARY 1967 to 25 JANUARY 1968

Performed for
GEORGE C. MARSHALL SPACE FLIGHT CENTER
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
HUNTSVILLE, ALABAMA

ABSTRACT

This report describes the work and results on a learning program for the numerical integration of systems of ordinary differential equations. (This is a continuation of work described in Gallaher, L. J., et al, "Study of the Methods for the Numerical Solution of Ordinary Differential Equations," Final Report Project A-831, Contract NAS8-20014, Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Georgia, (1967).)

The computer program was designed to incorporate the following features:

- a) meet a user specified accuracy, b) be problem independent (i.e. any number or degree of coupled differential equations), c) be self optimizing with respect to step size, order and integration methods used, d) exhibit learning so as to use a past performance history to determine methods and orders to be used.

The integration methods used are: a) the method of Adams, Bashforth and Moulton, b) Butcher's formulas for the Stetter-Gragg-Butcher method, c) Cowell's method of Nth order differences, d) Shank's formulas for the Runge-Kutta method.

The programs described here are written in both single (11 decimal place) and double (22 decimal place) precision Algol for the B-5500. An executive procedure acts in an administrative and bookkeeping capacity for the various integration methods. This executive procedure keeps the performance histories according to problem types, determines performance effectiveness of the methods and orders, and chooses those to be used.

Several kinds of problems and a large range of accuracies were used to exercise the program with the following results:

All methods perform well with no method or order always being exceptionally superior to the others.

For different kinds of problems and accuracies different methods and orders prove more effective.

Learning takes place in a satisfactory manner.

The program makes an effective general library procedure for integrating systems of ordinary differential equations.

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. INTEGRATION METHODS	5
A. The Method of Adams, Bashforth and Moulton	5
1. Description of the Method	5
2. The Computer Procedure	7
2.1 Parameters and Variables	8
2.2 The F Procedure	10
2.3 Orders Available	11
2.4 Starting an Integration	11
2.5 Error Estimates and Step Size Control	12
2.6 Finishing an Integration	13
3. Flow Diagram and Program Listing	14
4. Results and Conclusions	14
B. The Method of Stetter, Gragg, and Butcher	23
1. Description of the Method	23
2. The Computer Program	25
3. Flow Diagram and Program Listing	31
4. Results and Conclusions	31
C. The Cowell Method	43
1. Description of the Method	43
2. The Computer Program	45
3. Flow Diagram and Program Listing	54
4. Results and Conclusions	54

TABLE OF CONTENTS (Continued)

	Page
D. The Runge-Kutta-Shanks Method	69
1. Introduction	69
2. Description of the Method	69
3. The Computer Procedure	70
3.1 Error Estimates and Step Size Control	71
3.2 Input and Output of the Procedure	71
4. Flow Diagram and Program Listing	73
5. Results and Conclusions	73
E. The General Multistep Method Starting Procedure	84
1. Introduction	84
2. Description of the Procedure	84
3. Flow Diagram and Program Listing	87
III. THE EXECUTIVE PROCEDURE	99
A. Introduction	99
B. The Selection Process	100
C. Organization of the History File	102
D. Inputs to the Executive Procedure	106
1. The Single-Precision Program	106
2. The Double-Precision Program	107
E. Updating of the History File and Forgetting	108
F. Reading of Coefficients and History Files	109
G. Outputs of the Executive Procedure	110
H. Flow Diagram and Program Listing	111

TABLE OF CONTENTS (Continued)

	Page
IV. AUXILIARY PROGRAMS	149
A. The Coefficient File	149
B. Setting Up History Files	149
C. Printing of History Files	150
V. RESULTS AND CONCLUSIONS	187
A. Applications	187
B. Results	187
C. Conclusions	192
D. Recommendations for Further Study	192
VI. BIBLIOGRAPHY	195

LIST OF FIGURES

	Page
1. Flow Diagram for the Adams Method	16
2. Flow Diagram for the Stetter-Cragg-Butcher Method	32
3. Flow Diagram for the Cowell Method	56
4. Flow Diagram for the Runge-Kutta-Shanks Procedure	74
5. Flow Diagram for the General Multistep Method Starting Procedure	88
6. Flow Diagram for the Executive Procedure	112

LIST OF TABLES

	Page
I. ORGANIZATION OF CUMULATIVE WINNING AND LOSING TIMES BY METHOD AND ORDER	105
II. ORGANIZATION OF CUMULATIVE WINNING AND LOSING TIMES BY METHOD VS. METHOD	105
III. EXAMPLE HISTORY OF COMPETITIONS BETWEEN METHODS	188
IV. EXAMPLE HISTORY OF COMPETITIONS BETWEEN METHODS	189
V. EXAMPLE HISTORY OF COMPETITIONS BETWEEN METHODS	190
VI. EXAMPLE HISTORY OF COMPETITIONS BETWEEN METHODS	191

I. INTRODUCTION

The goal was set of writing a computer program for the integration of systems of ordinary differential equations (initial value problems), characterized by the following specifications:

- a) The integration must meet a (user's) specified accuracy.
- b) The procedure will be problem independent and applicable to the integration of any degree or number of coupled differential equations.
- c) The step size, order, and method of integration are to be chosen by the procedure so as to be optimum; that is, to minimize the computation time while meeting the accuracy requirements.
- d) The procedure will have built-in learning so that it can use its experience from one call to the next to decide on the method and order to be used. The procedure will be self-modifying.

In previous work under this contract [11, 21] a program attempting to meet these specifications was written in single-precision (11 decimal place) B-5500 ALGOL. Evaluation of this program indicated a high degree of success and encouraged further work along this line. This report describes the extensions and improvements in this program and further evaluation of the effectiveness of this approach to integrating differential equations.

Several major extensions and some minor changes were made in the program described in [21]. One change was the introduction of problem classification by the program and the maintenance of separate histories for each problem type. Another was the extension of the program to double-precision arithmetic (22 decimal places). This made possible the effective use of higher-order methods at smaller error tolerances. Both single and

double-precision programs are described here.

The methods used are as follows:

- (1) The Adams-Bashforth-Moulton method,
- (2) The Stetter-Gragg-Butcher method,
- (3) Cowell's method of constant Nth order differences,
- (4) The Runge-Kutta-Shanks method.

With each of these methods, four different orders are used. A history file is kept showing the past performance scores of each method and order by problem type and is used to select which methods and orders are to be employed.

The program works in the following way. When a call is made on the integration procedure, the first part of the interval is integrated by one method for each of two different orders, and the times taken by each recorded. A second part is integrated by another method, also for two different orders, and the times recorded. The winners then compete against each other over another part of the interval. That is, the faster order of the first method and the faster order of the second method are both used to integrate the next part of the interval, and the times taken by each recorded. The faster method of these two is then presumably the best (fastest) of the four tried, and it is used alone to integrate over the final portion of the interval. All of the times measured above are then logged in a cumulative history file, according to problem type with the winners and losers noted.

These history files are used as the basis for selecting which methods and orders are chosen each time.

The first of the two methods is chosen at random (using a random number generator) from among the three most effective available. The second method is chosen to be the method showing the best history of success

among the other methods, with the cumulative history file for that problem type used to determine the degree of success. Then within each method the same kind of selection process with respect to orders is used. That is, the first order is chosen at random from the three most effective and the second order is chosen on the basis of which of the others has been the most successful (fastest running) order of that method. Thus it is seen that the past performance of the different methods and orders influences the choice of which are allowed to compete, such that the more successful have a higher probability of being selected.

In using time as the sole estimate of performance efficiency, it is assumed that all orders and methods have satisfactorily met the accuracy requirements. The accuracy requirements of each method are met by controlling step size and making error estimates at each step. The method of error estimate is different for the different methods. In the Runge-Kutta single step method, the error is estimated by taking two half steps and then a whole step. In the Adams and Butcher methods the difference between predictor and corrector is used. In the Cowell method a mid-range formula is used.

One further feature introduced into the learning process is the gradual "forgetting" of events in the more distant past. This causes the events in the distant past to have less influence than those more recent in determining the score or performance figure of an order and method.

Three types of problems were used to exercise the integration procedures:

First, the Arenstorf type orbits of the restricted three-body problem (four equations).

Second, the system of linear differential equations associated with the Fourier transforms (20 to 40 equations).

Third, the system of linear differential equations obtained from a discretization of the partial differential equation for the vibrating string (50 to 100 equations).

The first of these is characterized by the necessity of frequent step-size change. The other two have no need for step-size change once the correct step is found.

Results of running with a variety of accuracies and problems mentioned above show that no particular method is exceptionally superior to any other. All methods performed well and, for different problem types, different methods show up more successfully. For example, the Runge-Kutta method was most successful at large error tolerances where frequent step size changes were required, but the multistep methods performed better where long runs of uniform step size were appropriate and for small error tolerances.

For accuracies obtainable in single precision the lower orders were more effective and, at the high accuracies (small error tolerances) obtainable in double precision, the higher orders were more effective. For a given accuracy one particular method and order usually dominated, but which one depended on the accuracy asked and on the problem type.

The results justify the conclusion that the present program would be suitable and effective as general library programs for integrating systems of differential equations.

II. INTEGRATION METHODS

A. The Method of Adams, Bashforth and Moulton

1. Description of the Method

The method investigated consists of the combination of two different versions of the method of Adams into a predictor-corrector system [5]. The use of this system to obtain numerical solutions to a set of simultaneous differential equations with given initial conditions is independent both of the number of equations in the set to be solved and of the orders of the individual equations in the set; provided, however, that each equation of order m is expressed as a set of m coupled first order equations.

In general then, one deals with the system of equations

$$\vec{y}'(x) \equiv \frac{d}{dx} \vec{y}(x) = \vec{f}(x, \vec{y}(x)), \quad (1-1)$$

where \vec{y}' , \vec{y} , and \vec{f} are vectors, each having a number of components, N , equal to $\sum_{i=1}^k m_i$, where k is the number of equations in the set to be solved, and the m_i are their individual orders.

This vector differential equation is equivalent to the integral equation

$$\vec{y}(x+h) = \vec{y}(x) + \int_x^{x+h} \vec{f}(t, \vec{y}(t)) dt. \quad (1-2)$$

At the point $x = x_q \equiv x_{q-1} + h$, this integral is approximated first by

$$\vec{y}_q^{(0)} = \vec{y}_{q-1} + h \sum_{\mu=0}^{q-1} \beta_{q-1, q-1-\mu} \vec{f}_\mu \quad (1-3a)$$

and then repeatedly by

$$\begin{aligned}\vec{y}_q^{(v+1)} &= h\beta_{q,0}^* \vec{f}(x_q, \vec{y}^{(v)}(x_q)) + h \sum_{\mu=0}^{q-1} \beta_{q,q-\mu}^* \vec{f}_\mu + \vec{y}_{q-1} \\ &\equiv h\beta_{q,0}^* \vec{f}^{(v)} + \vec{C}, \quad v = 0, 1, 2, \dots\end{aligned}\quad (1-3b)$$

Formula (1-3a) is called

the Adams-Bashforth predictor equation, and formula (1-3b) is the Adams-Moulton corrector.

The coefficients $\beta_{q\rho}$ and $\beta_{q\rho}^*$ are derived by the equivalent of integrating Lagrangian polynomials fitted to \vec{f} , but are independent of both \vec{f} and h . The polynomial for the predictor is of degree $q-1$ passing through the q points $\vec{f}_0, \vec{f}_1, \dots, \vec{f}_{q-1}$, while that for the corrector is of degree q passing through the $q+1$ points $\vec{f}_0, \vec{f}_1, \dots, \vec{f}_q$.

An explicit formula for the $\beta_{q\rho}$

$$\beta_{q\rho} = (-1)^\rho \left\{ \binom{\rho}{\rho} \gamma_\rho + \binom{\rho+1}{\rho} \gamma_{\rho+1} + \dots + \binom{q}{\rho} \gamma_q \right\}, \quad \begin{matrix} q = 0, 1, 2, \dots \\ \rho = 0, 1, \dots, q \end{matrix}$$

where the $\binom{\rho+i}{\rho}$ represent binomial coefficients and the γ_ρ are found by the recursion relation

$$\gamma_\rho + 1/2 \gamma_{\rho-1} + \dots + \frac{1}{\rho+1} \gamma_0 = 1, \quad \rho = 0, 1, 2, \dots$$

An explicit formula for the $\beta_{q\rho}^*$ is

$$\beta_{q\rho}^* = (-1)^\rho \left\{ \binom{\rho}{\rho} \gamma_\rho^* + \binom{\rho+1}{\rho} \gamma_{\rho+1}^* + \dots + \binom{q}{\rho} \gamma_q^* \right\}, \quad \begin{matrix} q = 0, 1, 2, \dots \\ \rho = 0, 1, \dots, q \end{matrix}$$

where $\gamma_0^* = 1$ and $\gamma_\rho^* = \gamma_\rho - \gamma_{\rho-1}$, $\rho = 1, 2, 3, \dots$

Bounds on the errors for the two approximations are the maximums within the interval $[x_0, x_q]$ of

$$\left| y_q h^{q+1} \frac{d^{q+1}}{dx^{q+1}} \vec{y} \right| \quad (\text{for Adams-Bashforth}) \quad (1-4a)$$

and of

$$\left| y_{q+1}^* h^{q+2} \frac{d^{q+2}}{dx^{q+2}} \vec{y} \right| \quad (\text{for Adams-Moulton}) \quad (1-4b)$$

and M , the order of the predictor-corrector system, is assumed to approximate that of the corrector, which is $q + 1$.

2. The Computer Procedure

Two computer procedures have been written to implement the Adams method, a single-precision version named ADAMS and a double-precision version named DADAMS. Only DADAMS will be described here. A description of procedure ADAMS can be found in [21].

Procedure DADAMS is written to be included in programs written in double precision for the Burroughs B-5500 computer. The language is Algol 60 augmented by additional features available in the Algol compiler for the B-5500. There are no unusual hardware requirements because all input and output to the procedure is under control of the including program through the formal parameter list. All variables not in the formal parameter list are local to the procedure, and no files are used by the procedure.

2.1 Parameters and Variables

The following lists of formal parameters and local variables will be useful in describing the operation of procedure DADAMS. In the remainder of this discussion the interchange of upper and lower case letters, necessitated by approximating the notation of [5] within the limited character set available to a computer, is straight-forward and will be done freely without further comment. For double-precision variables the identifier for only the most significant portion is given. The identifier for the least significant portion is the same, followed by an "L".

Formal Parameters

<u>Identifier</u>	<u>Type</u>	<u>Usage or Meaning</u>
N	Integer	The number of components in the vectors \vec{y} , EA, and ER.
XI	Double Real	Initial value of the independent variable.
XF	Double Real	Final value of the independent variable.
Y	Double Real Array	Current dependent variable vector. Contains initial values at entry and final values at exit.
F	Procedure	Calculates the vector $\vec{f}(x, \vec{y}(x))$.
P	Real	Power of C1 used in error control.
Q	Integer	Number of back \vec{f} points used in the approximating polynomials. One less than M, the order of the method.
DX	Real	Upper bound on the initial step size.
EA	Real Array	Absolute error bound vector.
ER	Real Array	Relative error bound vector.
ADAMSCOEFF	Double Real Array	Contains the $\beta_{q\rho}$, the $\beta_{q\rho}^*$, and $\left 1 - \gamma_{q+1}/\gamma_{q+1}^* \right $.

RKSFNEVAL	Integer	Function evaluations per step for procedures DSTART and DSHANKS.
RKORDER	Integer	Order of R.K.S. method to be used by DSTART and DSHANKS.
RKSCOEFF	Double Real Array	Coefficients for DSTART and DSHANKS. See the descriptions of DSTART and DSHANKS elsewhere in this report for details.
DSTART	Integer Procedure	Gives the necessary points for starting and restarting. Name contains the factor by which C1 is multiplied to coordinate step size between DSTART and DADAMS.
DSHANKS	Procedure	Used to complete fractional steps at the ends of intervals.
UEB	Boolean	True if the upper error bounds are to be used for corrector iteration control. False if the lower error bounds are to be used.

The procedures DSTART, DSHANKS, and F, as well as the coefficient arrays ADAMSCOEFF and RKSCOEFF, are not a part of the procedure DADAMS and must be included separately in all programs using DADAMS (see 2.2 and 2.3).

Local Variables

<u>Identifier</u>	<u>Type</u>	<u>Usage or Meaning</u>
X	Double Real	x_q , current value of the independent variable.
INTERV	Double Real	$X_F - X_I$, the interval of integration.
C1	Integer	Two to an integer power. Determines H.
H	Double Real	$INTERV / C1$, the current step size.
C2	Double Real	Number of steps of size H remaining from X to X_F .
GR	Real	$\left 1 - \gamma_{q+1} / \gamma_{q+1}^* \right $, used with CHANGE and ERROR. (Has a dummy lower half in ADAMSCOEFFL).

CHANGE	Real	Controls the number of iterations of the corrector equation. (Has a dummy lower half).
ERROR	Real	Controls the error and running time through the step size. (Has a dummy lower half).
FNU	Double Real Array	Holds the successive vectors $\vec{f}_q^{(v)}$ of (1-3b).
FH	Double Real Array	\vec{f} history vector. Contains $2q-1$ back points for each of the N components of \vec{f} (Two dimensions).
YP	Double Real Array	Predicted \vec{y}_q vector, $\vec{y}^{(p)}$, the $y_q^{(o)}$ of (1-3a).
YC	Double Real Array	Corrected \vec{y}_q vector, $\vec{y}^{(c)}$, the $\vec{y}_q^{(v+1)}$ of (1-3b).
C	Double Real Array	Constant part of $\vec{y}^{(v+1)}$ for all v , the \vec{C} of (1-3b).

All local arrays are dynamic with respect to N and Q and the \vec{FH} vector array is indexed cyclically to avoid moving large numbers of components. For further details consult the flow diagram and the listing of procedure DADAMS following this discussion.

2.2 The F Procedure

A procedure for calculating the vector $\vec{y}' = \vec{f}(x, \vec{y}(x))$ must be included global to a call for procedure DADAMS for each set of differential equations to be solved by a program using DADAMS. This procedure is called by DADAMS as the formal parameter F and must itself have the following formal parameter list:

<u>Identifier</u>	<u>Type</u>	<u>Usage or Meaning</u>
N	Integer	Number of components in the vectors \vec{YV} and \vec{FV} .
X	Double Real	Current value of the independent variable.
YV	Double Real Array	Current dependent variable vector (input).
FV	Double Real Array	\vec{F} value vector (output).

N and X may be called by value. The arrays YV and FV are one-dimensional starting at zero and must be called by name.

2.3 Orders Available

Procedure DADAMS is written to be completely general with regard to order, and any order may be used if the necessary coefficients are placed in the ADAMSCOEFF array. For a given order $M = q + 1$, there are $2q + 2 = 2M$ coefficients which should appear in the array beginning at position zero in the following order:

$$\beta_{q-1,q-1}, \beta_{q-1,q-2}, \dots, \beta_{q-1,0}, \beta_{q,q}^*, \beta_{q,q-1}^*, \dots, \beta_{q,0}^*, \left| 1 - \gamma_{q+1} / \gamma_{q+1}^* \right|.$$

2.4 Starting an Integration

Since the Adams method is a multistep method, it cannot start itself but must rely on a starting procedure that will supply a current \vec{y} point and at least $q-1$ \vec{f} points which, together with a given initial \vec{f} point, comprise a history upon which it can build. The starting procedure used here is the Runge-Kutta-Shanks procedure DSTART, described elsewhere in this report. The number of function evaluations per step and the order of Runge-Kutta-Shanks method used by DSTART may be varied at will by the user through the formal parameters of DADAMS. This allows achievement of optimum compatibility with the order of Adams method being used for each given set of differential equations being solved.

Initial step size is determined by the formal parameter DX. The initial trial start will be made with a step $H = \text{INTERV} / C1$, where $C1$ is set to the smallest integer power of two such that $|H| \leq |DX|$ and $|H| \leq |\text{INTERV}|/Q$. This causes procedure DADAMS to take at least one step

after starting regardless of the magnitude of DX. If procedure DSTART cannot meet the error requirements at the initial H, it doubles C1 repeatedly until these requirements can be met.

2.5 Error Estimates and Step Size Control

To minimize running time without introducing errors intolerably large, the error in each component of the final \vec{Y} vector is controlled through the use of the formal parameters \vec{EA} and \vec{ER} . \vec{EA} specifies the maximum allowable absolute magnitude of the error in each component of \vec{Y} , and \vec{ER} specifies the maximum allowable relative magnitude. These two error control vectors are used in conjunction with the quantity $GR = \left| 1 - \gamma_q / \gamma_{q+1}^* \right|$, which is derived from the bounds (1-4), and a parameter P, chosen from the interval $[1/2, 1]$ by empirical determination of the randomness of the round-off error in a particular set of differential equations. (P = 1/2 corresponds to totally random error and P = 1 corresponds to totally additive linear error). In practice γ_{q+1} has been used in GR instead of γ_q to be conservative because the quantity being controlled is only an estimate of the true error.

All control of corrector iterations and step size is done in single precision using the most significant portion of the variables. To make a trial step, procedure DADAMS increases X by the current H and calculates $\vec{y}^{(p)}$, the $\vec{y}_q^{(o)}$ of (1-3a), together with the \vec{C} of (1-3b). It then calculates $\vec{f}_q^{(o)} = \vec{f}(x, \vec{y}^{(p)})$ and $\vec{y}^{(c)}$, the $y_q^{(v+1)}$ of (1-3b).

At each corrector iteration, v, the vector $\vec{CHANGE} \equiv \vec{y}^{(v+1)} - \vec{y}^{(v)}$ is tested in one of two ways according to the setting of the Boolean parameter UEB. When UEB is true, if every component of \vec{CHANGE} is less than the corresponding component of either $\left| \frac{\vec{EA} \cdot GR}{C1^P} \right|$ or $\left| \frac{\vec{ER} \cdot GR}{C1^P} \cdot \vec{y}^{(c)} \right|$, then $\vec{f}^{(v)}$ and $\vec{y}^{(c)}$

are retained and the corrector iterations are terminated. When UEB is false every component of $\vec{\text{CHANGE}}$ must be smaller than the corresponding component of either $\left| \frac{\vec{\text{EA}} \cdot \text{GR}}{\text{Cl}^{\text{P}} \cdot 2^{\text{Q}+5}} \right|$ or $\left| \frac{\vec{\text{ER}} \cdot \text{GR}}{\text{Cl}^{\text{P}} \cdot 2^{\text{Q}+5}} \cdot \vec{\text{f}}^{(\text{v})} \right|$ before $\vec{\text{f}}^{(\text{v})}$ and $\vec{\text{y}}^{(\text{c})}$ are accepted as final. For either setting of UEB, if any component of $\vec{\text{CHANGE}}$ fails to meet the proper criterion, a new $\vec{\text{f}}^{(\text{v})}$, $\vec{\text{y}}^{(\text{c})}$, and $\vec{\text{CHANGE}}$ are computed and the test is then repeated. Whenever UEB is true the retained $\vec{\text{f}}^{(\text{v})}$ and $\vec{\text{y}}^{(\text{c})}$ are corrected one additional time before the vector $\vec{\text{ERROR}}$ is formed.

The estimated error vector $\vec{\text{ERROR}}$ is defined to be $\left| \vec{\text{y}}^{(\text{c})} - \vec{\text{y}}^{(\text{p})} \right|$.

If any component of $\vec{\text{ERROR}}$ is larger than the corresponding components of both $\left| \frac{\vec{\text{EA}} \cdot \text{GR}}{\text{Cl}^{\text{P}}} \right|$ and $\left| \frac{\vec{\text{ER}} \cdot \text{GR}}{\text{Cl}^{\text{P}}} \cdot \vec{\text{y}}^{(\text{c})} \right|$, the step size is halved and q-1 new $\vec{\text{f}}$ points and a new current $\vec{\text{y}}$ are obtained from procedure DSTART. If it is not necessary to halve the step size, $\vec{\text{y}}^{(\text{c})}$ becomes the new $\vec{\text{y}}$. If every component of $\vec{\text{ERROR}}$ is smaller for three consecutive steps than the corresponding components of both $\left| \frac{\vec{\text{EA}} \cdot \text{GR}}{\text{Cl}^{\text{P}} \cdot 2^{\text{Q}+5}} \right|$ and $\left| \frac{\vec{\text{ER}} \cdot \text{GR}}{\text{Cl}^{\text{P}} \cdot 2^{\text{Q}+5}} \cdot \vec{\text{y}}^{(\text{c})} \right|$, then if there are at least 2q-1 back points in the FH vector and there are at least two more steps of the current size necessary to reach XF, the step size is doubled before the next trial step. If it is not necessary either to halve or to double the step size, X is increased by H and a new trial step is made.

2.6 Finishing an Integration

Procedure DADAMS continues as described until XF is reached unless repeated halvings and doublings of the step size bring the independent variable to within a fraction of a single step of XF. When this occurs the fractional step is completed by the Runge-Kutta-Shanks procedure DSHANKS, described elsewhere in this report. The order of Runge-Kutta-Shanks method

and the number of function evaluations per step used here will be the same for a given integration as those used by procedure DSTART.

3. Flow Diagram and Program Listing

Figure 1 is the flow diagram for the method of Adams, Bashforth and Moulton. The program listing follows at the end of this section.

4. Results and Conclusions

The two methods of corrector iteration control included in DADAMS were selected from a set of ten on the basis of results of a small number of test cases. Performance of these two methods was then compared in a large number of cases involving four different sets of equations, values of q between 10 and 15, and accuracies asked between 10^{-10} and 10^{-20} . Several fairly clear results were obtained.

(1) The step size controls always produced at least the accuracy asked, using both methods of iteration control, except with one set of equations believed to have extremely additive nonlinear error buildup. With this set of equations it was necessary to request from 1.3 to 16.3 times (not decades) the accuracy desired throughout the range obtainable, even with the parameter p set at one. To obtain these results it was necessary to choose a reasonable value of p ($p \geq 1/2$) and a reasonable error to control (relative error for rapidly increasing integrals and absolute error for rapidly decreasing integrals).

(2) When plotted on double log paper, accuracy obtained vs accuracy asked was slightly erratic but roughly parallel to the 45° reference line across many decades for all cases plotted.

(3) Except for a small percentage of cases distributed mostly at random, the values of the parameters for procedure DSTART which resulted in the least function evaluations to obtain a given accuracy were RKSORDER = 7 and RKSFNEVAL = 9; however, the values RKSORDER = 8 and RKSFNEVAL = 12 did produce greater efficiency at an accuracy asked of 10^{-20} for all values of q on one rapidly decreasing integral.

(4) The values of q giving the greatest accuracy obtained for a given number of function evaluations on all sets of equations tested were 10, 11, and occasionally 12. Higher q 's were always less efficient.

(5) Plots of function evaluations against accuracy obtained showed little difference between the two methods of corrector iteration control on two sets of equations. On a third set of equations one method of control showed a much steeper slope than the other for each value of q , giving it a higher efficiency at lower accuracies and a lower efficiency at higher accuracies. On the fourth set of equations this phenomenon reappeared, but in this case the opposite method had the steeper slope. In each of these cases the absolute error was being controlled. There was also some crossing of the functions vs accuracy plots for different values of q for each method of iteration control on both sets of equations.

Users preparing to run a large number of cases with a single set of equations should often find it helpful to plot the results of methodical variations in the parameters of DADAMS. The results reported here did not appear upon examination of the table of test cases used to select the two methods of corrector iteration control, but emerged only after the plotting of a large number of cases run for evaluation of the two methods chosen.

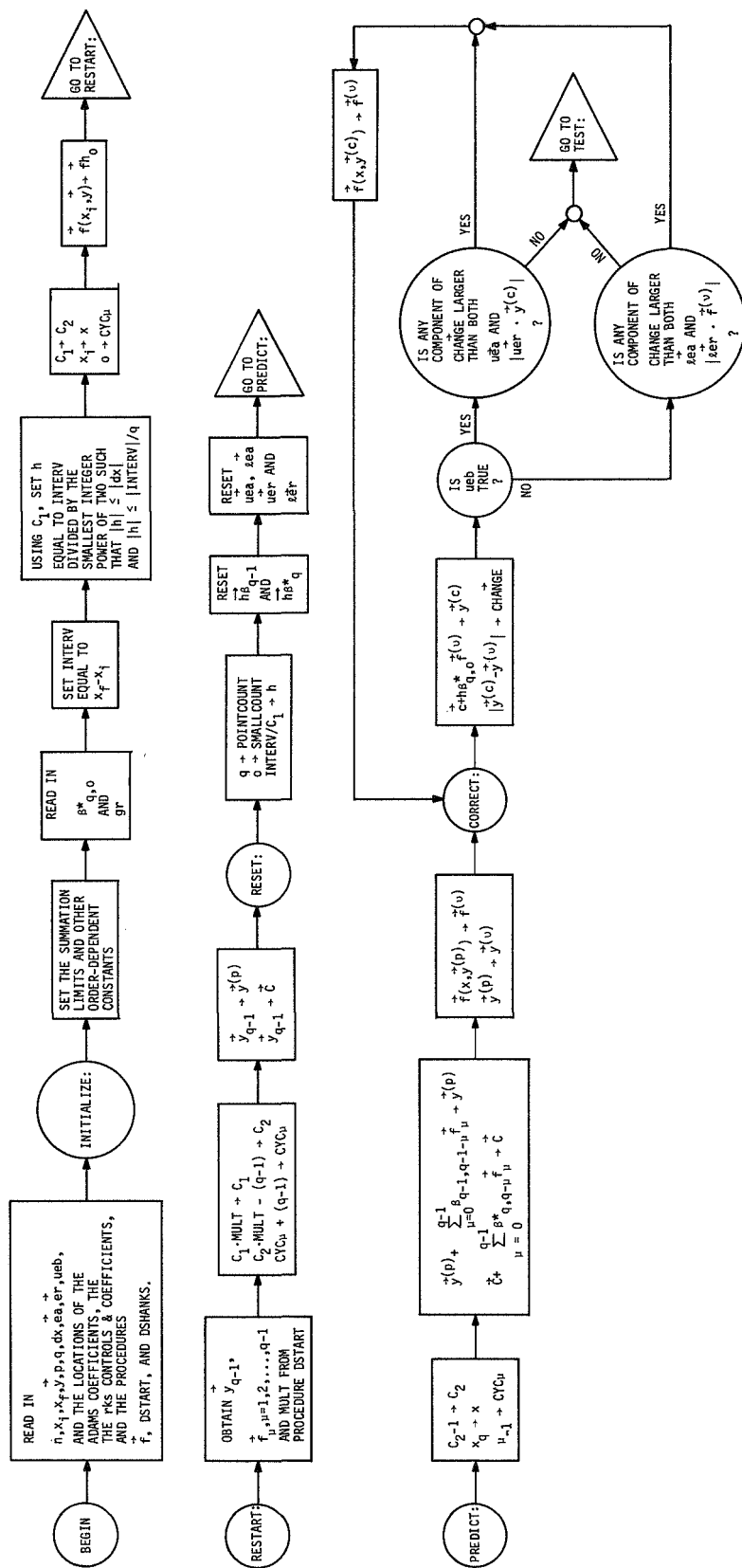


Figure 1. Flow Diagram for the Adams Method.

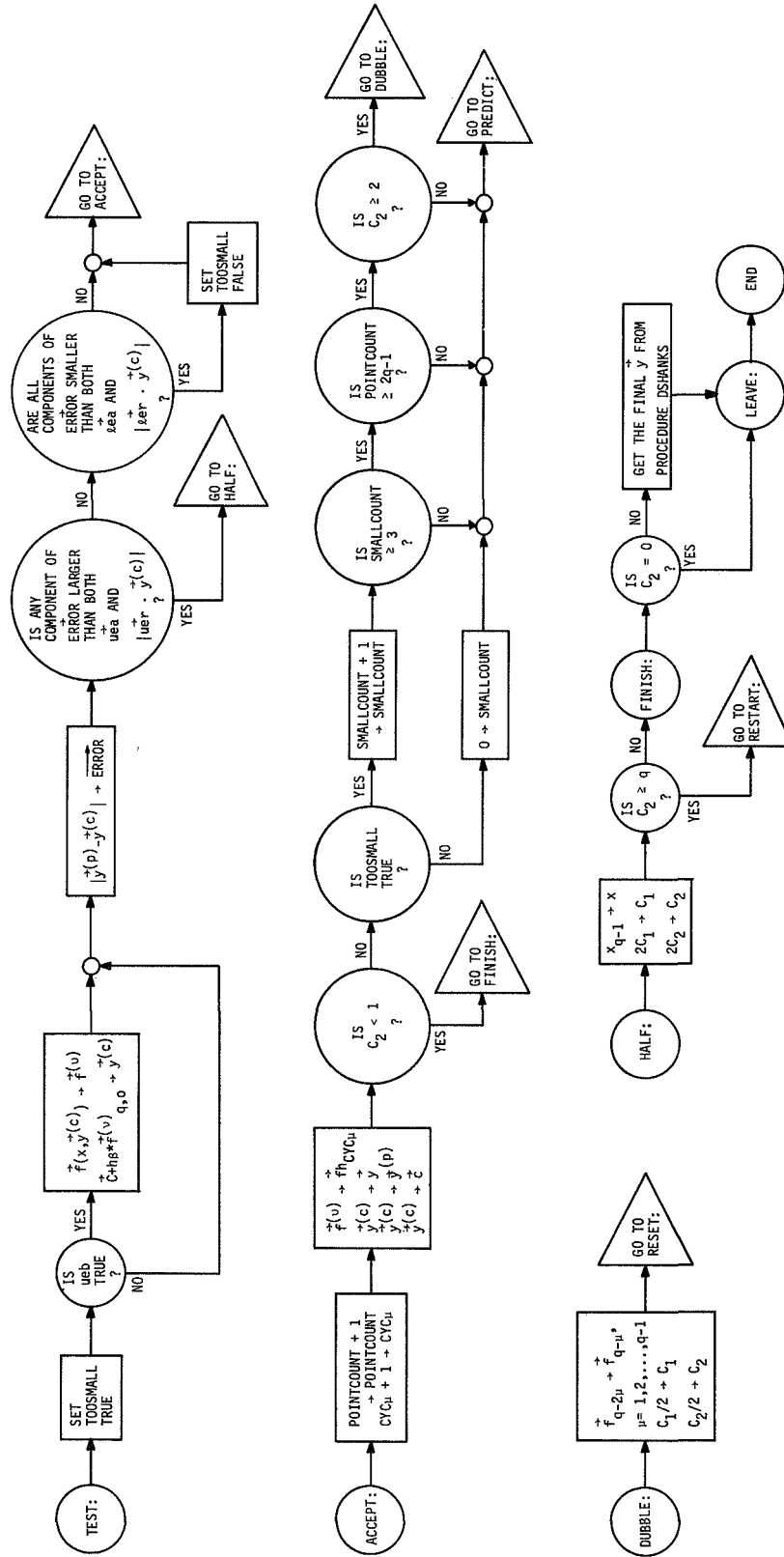


Figure 1 (Continued) Flow Diagram for the Adams Method.

Repeated step size expansions and contractions by over 1000 to 1 occurred on one set of equations and it is probable that for many sets of equations slight changes in control logic can produce considerable changes in efficiency.

```

PROCEDURE DADAMS (N,XI,XIL,XF,XFL,Y,YL,F,P,Q,DX,EA,ER,ADAMSCOEFF, 00000000
ADAMSCOEFFL,RKSFNEVAL,RKSORDER,RKSCOEFF,RKSCOEFFL,DSTART,DSHANKS,UEB) ;00001000
VALUE N,XI,XIL,XF,XFL,P,Q,DX,RKSFNEVAL,RKSORDER,UEB ; 00002000
REAL XI,XIL,XF,XFL,P,DX ; 00003000
INTEGER N,Q,RKSFNEVAL,RKSORDER ; 00004000
REAL ARRAY Y,YL,EA,ER,ADAMSCOEFF,ADAMSCOEFFL,RKSCOEFF,RKSCOEFFL [0] ; 00005000
PROCEDURE F,DSHANKS ; 00006000
INTEGER PROCEDURE DSTART ; 00007000
BOOLEAN UEB ; 00008000
00009000
00010000
00011000
00012000
00013000
00014000
00015000
00016000
00017000
00018000
00019000
00020000
00021000
00022000
00023000
00024000
00025000
00026000
00027000
00028000
00029000
00030000
00031000
00032000
00033000
00034000
00035000
00036000
00037000
00038000
00039000

BEGIN
  DEFINE ALLMU = FOR MU ← 0 STEP 1 UNTIL QMINUS1 DO # , ALLI = FOR I ←
  STEP 1 UNTIL N DO # ;
  LABEL INITIALIZE,RESTART,RESET,PREDICT,CORRECT,TEST,ACCEPT, DUBBLE,
  HALF,FINISH,LEAVE ;
  REAL ARRAY FH,FHL [0:2×Q-2:Q:N], HB,HBL,HBS,HBSL [0:Q-1], C,CL,YC,YCL
  ,YP,YPL,FNU,FNUL,LEA,LER,UEA,UER [0:N] ;
  REAL H,HL,X,XL,C2,C2L,YCI,YCIL,BSQZ,BSQZL,FMUI,FMUIL,HBMU,HBMUL, HBSMU,00017000
  ,HBSMUL,HBSQZ,HBSQZL,INTERV,INTERVL,CHANGE,CHANGE, ERROR,ERRORL,CU,GR,00018000
  ,FNUI,C2MQP5 ;
  INTEGER I,J,K,C1,MU,MULT,CYCMU,QT2M1,QT2M2,QMINUS1,QTIMES2, POINTCOUNT,00020000
  ,SMALLCOUNT ;
  BOOLEAN TOOSMALL ;
  INITIALIZE : QMINUS1 ← Q - 1 ;
  QTIMES2 ← Q + Q ;
  QT2M1 ← QTIMES2 - 1 ;
  QT2M2 ← QT2M1 - 1 ;
  C2MQP5 ← 2*(-(Q+5)) ;
  DOUBLE (ADAMSCOEFF[QTIMES2],ADAMSCOEFFL[QTIMES2],←BSQZ,BSQZL) ;
  GR ← ADAMSCOEFF [QTIMES2+1] ;
  DOUBLE (XF,XFL,XI,XIL,←INTERV,INTERVL) ;
  DOUBLE (INTERV,INTERVL,←H,HL) ;
  DX ← ABS (DX) ;
  C1 ← 1 ;
  WHILE ABS (H) > DX OR C1 < Q DO
  BEGIN
    C1 ← C1 + C1 ;
    DOUBLE (INTERV,INTERVL,C1,0,←H,HL) ;
  END ;

```

```

DOUBLE (C1,0,←C2,C2L) ;
F (N,XI,XIL,Y,YL,FH [CYCMU←0,*,1,FHL [0,*]]) ;
DOUBLE (XI,XIL,←X,XL) ;
RESTART : C1 ← C1XMULT ← DSTART (N,XI,XIL,XF,XFL,C1,EA,ER,F,QMINUS1,
,XL,Y,YL,FH,FHL,FH,FHL,Y,YPL,CYCMU, QT2M1,0,P,RKSFNEVAL,RKSCDEFF,
RKSCDEFFL) ;
DOUBLE (C2,C2L,MULT,0,X,QMINUS1,0,←C2,C2L) ;
CYCMU ← (CYCHU+QMINUS1) MOD QT2M1 ;
ALLI DOUBLE (CII,←YPI,I),CLII,←YPL[II],←YII,YL[II] ;
RESET : POINTCOUNT ← 0 ;
SMALLCOUNT ← 0 ;
DOUBLE (INTERV,INTERVL,C1,0,/,←H,HL) ;
ALLMU
BEGIN
DOUBLE (ADAMSCOEFF [MU],ADAMSCOEFFL [MU],H,HL,X,←HB [MU],HBL [MU])
;
DOUBLE (ADAMSCOEFF[MU+Q],ADAMSCOEFFL[MU+Q],H,HL,X,←HBS[MU],HBSL[MU]
) ;
END ;
DOUBLE (BSQZ,BSQZL,H,HL,X,←HBSQZ,HBSQZL) ;
CU ← (C1+(-P))XGR ;
ALLI
BEGIN
LEA [II] ← (UEA [II] ← EA [II]XCU)XC2MQP5 ;
LER [II] ← (UER [II] ← ER [II]XCU)XC2MQP5 ;
END ;
PREDICT : DOUBLE (C2,C2L,1,0,←C2,C2L) ;
DOUBLE (XF,XFL,H,HL,C2,C2L,X,←X,XL) ;
CYCMU ← (CYCMU+QMINUS1) MOD QT2M1 ;
ALLMU
BEGIN
CYCMU ← (CYCMU+1) MOD QT2M1 ;
DOUBLE (HB [MU],HBL [MU],←HBMU ,HBMUL ) ;
DOUBLE (HBS[MU],HBSL[MU],←HBSMU,HBSMUL) ;
ALLI
BEGIN
DOUBLE (FMUI←FHI(CYCMU,I),FMUIL←FHL(CYCMU,I),HBMU,HBMUL,X, YPI[I],
YPL[I],←YPI[I],YPL[I]) ;

```

```

00040000
00041000
00042000
00043000
00044000
00045000
00046000
00047000
00048000
00049000
00050000
00051000
00052000
00053000
00054000
00055000
00056000
00057000
00058000
00059000
00060000
00061000
00062000
00063000
00064000
00065000
00066000
00067000
00068000
00069000
00070000
00071000
00072000
00073000
00074000
00075000
00076000
00077000
00078000
00079000

```

```

DOUBLE (C[I],CL[I],FMUI,FMUIL,HBSMU,HBSMUL,X,+,+,C[I],CL[I]);
00080000
END ;
00081000
00082000
00083000
00084000
00085000
00086000
00087000
00088000
00089000
DOUBLE (FNUI←FNU[I],FNUL[I],HBSQZ,HBSQZL,X,C[I],CL[I],+,+,YCI,YCIL)
;
00090000
DOUBLE (YCI,YCL[I],YC[I]←YCI,YCL[I]←YCIL,+,+,CHANGE,CHANGE);
00091000
IF CHANGE ← ABS(CHANGE) > (IF UEB THEN UEA[I] ELSE LEA[I]) THEN IF
00092000
CHANGE > ABS (IF UEB THEN UER[I]×YCI ELSE LER[I]×FNUI) THEN
00093000
BEGIN
00094000
WHILE I < N DO DOUBLE (FNUI←I+1,FNUL[I],HBSQZ,HBSQZL,X,CL[I],CL
00095000
[I],+,+,YC[I],YCL[I]);
00096000
F (N,X,XL,YC,YCL,FNU,FNU);
00097000
GO TO CORRECT ;
00098000
00099000
00100000
00101000
00102000
00103000
00104000
00105000
00106000
00107000
IF UEB THEN DOUBLE (FNU[I],FNUL[I],HBSQZ,HBSQZL,X,C[I],CL[I],+,+,YC
00108000
[I],YCL[I]);
00109000
DOUBLE (YCI←YC[I],YCL[I],YPL[I],YPL[I],+,+,ERROR,ERRORL);
00110000
IF ERROR ← ABS (ERROR) > UEA [I] THEN IF ERROR > ABS (UER[I]×YCI)
00111000
THEN GO TO HALF ;
00112000
IF TOOSMALL THEN IF ERROR > LEA [I] THEN IF ERROR > ABS (LER[I]×YCI)
00113000
THEN TOOSMALL ← FALSE ;
00114000
00115000
00116000
00117000
00118000
00119000
END ;
ACCEPT : POINTCOUNT ← POINTCOUNT + 1 ;
CYCMU ← (CYCMU+1) MOD QT2M1 ;
ALLI
BEGIN

```

```

DOUBLE (FNUL[I],FNUL[I],←,FH[C,YCMU,I],FHL[C,YCMU,I]) ;
DOUBLE (C[I]←Y2[I]←YC[I],CL[I]←YPL[I]←YCL[I],←,Y[I],YL[I]) ;
END ;
IF C2 < 1.0 THEN GO TO FINISH ;
IF TOO SMALL THEN
BEGIN
  IF SMALLCOUNT + SMALLCOUNT+1 ≥ 3 THEN IF POINTCOUNT ≥ QT2M1 THEN IF
    C2 ≥ 2.0 THEN GO TO DOUBBLE ;
  END ELSE SMALLCOUNT ← 0 ;
  GO TO PREDICT ;
  DOUBBLE : K ← J ← CYCMU ;
  FOR MU ← 1 STEP 1 UNTIL QMINUS1 DO
  BEGIN
    IF J ← J-1 < 0 THEN J ← QT2M2 ;
    IF K ← K-2 < 0 THEN K ← K+QT2M1 ;
    ALLI DOUBLE (FH[K,I],FHL[K,I],←,FH[J,I],FHL[J,I]) ;
  END ;
  C1 ← C1 DIV 2 ;
  DOUBBLE (C2,C2L,←,2.0,←,←,C2,C2L) ;
  GO TO RESET ;
  HALF : DOUBBLE (C2,C2L,←,1.0,←,←,C2,C2L) ;
  DOUBBLE (XF,XFL,H,HL,C2,C2L,x,←,←,X,XL) ;
  C1 ← C1 + C1 ;
  DOUBBLE (C2,C2L,C2,C2L,←,←,C2,C2L) ;
  IF C2 ≥ Q THEN GO TO RESTART ;
  FINISH : IF C2 = 0 THEN GO TO LEAVE ;
  DSHANKS (N,X,XL,XF,XFL,Y,YL,F,RKSFNEVAL,RKORDER,RKSCOEFF,RKSCOEFFL,
    P,EA,ER,XF=X) ;
  LEAVE :
  END DADAMS ;

```

```

00120000
00121000
00122000
00123000
00124000
00125000
00126000
00127000
00128000
00129000
00130000
00131000
00132000
00133000
00134000
00135000
00136000
00137000
00138000
00139000
00140000
00141000
00142000
00143000
00144000
00145000
00146000
00147000
00148000
00149000
00150000
00151000
00152000

```

B. The Method of Stetter, Gragg, and Butcher

1. Description of the Method

The method of Stetter, Gragg, and Butcher, abbreviated Butcher's method in this report, is a multistep predictor-corrector method for the numerical solution of the first-order vector differential equation

$$\vec{y}'(x) = \frac{d}{dx} \vec{y}(x) = \vec{F}(x, \vec{y}(x)), \quad \vec{y}(x_0) = \vec{y}_0 \quad (1-1)$$

A complete derivation and description of Butcher's method can be found in [13] and [21]; only the essential formulas are included here.

The following notation is adopted. Let k be a positive integer, h be the step size (assumed to be constant over some set of calculations),

$$x_n = x_0 + n h, \quad \vec{y}_n = \vec{y}(x_n), \quad \text{and} \quad \vec{f}_n = \vec{F}(x_n, \vec{y}_n).$$

Butcher's method consists of two predictor and one corrector formulas.

The first predictor formula is

$$\vec{y}_{n-\theta} = \sum_{j=1}^k A_j \vec{y}_{n-j} + h \sum_{j=1}^k B_j \vec{f}_{n-j}, \quad (1-2)$$

the second predictor formula is

$$\vec{y}_n = \sum_{j=1}^k a_j \vec{y}_{n-j} + h b \vec{f}_{n-\theta} + h \sum_{j=1}^k b_j \vec{f}_{n-j}, \quad (1-3)$$

and the corrector formula is

$$\vec{y}_n = \sum_{j=1}^k \alpha_j \vec{y}_{n-j} + h (\beta \vec{f}_{n-\theta} + \beta_0 \vec{f}_n) + h \sum_{j=1}^k \beta_j \vec{f}_{n-j}. \quad (1-4)$$

The predictor formula (1-2) gives $\vec{y}_{n-\theta}$ in terms of the y values and the function values at the k points previous to x ; for $0 < \theta < 1$, $\vec{y}_{n-\theta}$ is the value at a point between x_{n-1} and x_n . The predictor formula (1-3) gives \vec{y}_n in terms of the y values and the function values at the previous k points as well as the function value $\vec{f}_{n-\theta}$. The corrector formula (1-4) gives \vec{y}_n in terms of the y values and the function values at the previous k points as well as the function values $\vec{f}_{n-\theta}$ and the function value \vec{f}_n obtained from the \vec{y}_n predicted by (1-3). Stable formulas (in the sense of Dahlquist [3], [4]) exist for $1 \leq k \leq 7$; the corrector is of order $2k + 1$ in these formulas. The coefficients for $1 \leq k \leq 3$, $\theta = 1/2$, and for $4 \leq k \leq 6$, $\theta = 1/3$, are given in [13].

For $k = 1$, Butcher's method is self-starting, for only \vec{y}_{n-1} and \vec{f}_{n-1} are needed to apply (1-2). For $k > 1$, however, it must be assumed that the values

$$\left\{ \vec{y}_{n-i} \right\}_{i=1}^k \quad \text{and} \quad \left\{ \vec{f}_{n-i} \right\}_{i=1}^k$$

have been obtained from some starting procedure or from previous calculations. (1-2) is applied to obtain $\vec{y}_{n-\theta}$, and $\vec{f}_{n-\theta}$ is then calculated. (1-3) is next applied to obtain \vec{y}_n , and \vec{f}_n is then calculated. Finally, (1-4) is applied once to obtain \vec{y}_n . The predicted value of \vec{y}_n , obtained from (1-3), is compared with the corrected value of \vec{y}_n , obtained from (1-4). If the two values are in sufficient agreement, the step is accepted; if not, the step is rejected.

Note that the corrector (1-4) is only applied once; repeated application of this corrector leads to less rather than more accuracy.

2. The Computer Program

The Butcher computer program is a Burroughs B-5500 ALGOL double-precision procedure whose declaration is as follows:

```
procedure butcher (n, xi, xil, xf, xfl, y, yl, f, ea, er, p, dx, rksfn,
                  rksorder, rkscoeff, rkscoeffl, boogerfactor, k,
                  butchercoeff, butchercoeffl, start, dshanks);
value n, xi, xil, xf, xfl, p, dx, rksfn, rksorder, k;
integer n, rksfn, rksorder, k;
real xi, xil, xf, xfl, p, dx, boogerfactor;
real array y, yl, ea, er, rkscoeff, rkscoeffl, butchercoeff,
          butchercoeffl [0];
procedure f, dshanks;
integer procedure start;
```

The parameters of the procedure are defined as follows:

n - the number of dependent variables in the vectors \vec{y} and \vec{f}
xi, xil - the high and low halves, respectively, of x_0 , the starting value of the independent variable x
xf, xfl - the high and low halves, respectively, of the final value of the independent variable x
y, yl - the arrays in which are located the high and low halves, respectively, of $\vec{y}_0 = \vec{y}(\underline{xi}, \underline{xil})$ upon entry and of $\vec{y}(\underline{xf}, \underline{xfl})$ upon exit
f - the double-precision procedure which computes $\vec{f} = \vec{f}(x, \vec{y})$
ea - the array containing the absolute error vector
er - the array containing the relative error vector
p - the exponent used in step-size control
dx - the suggested initial step size

rksfn - the number of function evaluations used in the Runge-Kutta-Shanks starting and closing procedure

rksorder - the order of the Runge-Kutta-Shanks closing procedure

rkscoeff, rkscoeffl - the arrays containing the high and low halves, respectively, of the Runge-Kutta-Shanks coefficients for the starting and closing procedures

boogerfactor - a fudge factor used in step-size control

k - the integer used in describing Butcher's method

butchercoeff, butchercoeffl - the arrays containing the high and low halves, respectively, of the Butcher coefficients

start - the double-precision starting procedure

dshanks - the double-precision closing procedure

The procedure performs the numerical integration of (1-1) in double precision from $x = \underline{x_i}$ to $x = \underline{x_f}$ (For the convenience of description, the low half of a variable is often not mentioned.). The step size h used is always the length of the interval $\underline{x_f} - \underline{x_i}$ divided by a power of 2. In order to avoid error build-up in the independent variable two counters, c1 and c2, are kept. c1 is always a positive, integral power of 2, and $h = (\underline{x_f} - \underline{x_i})/\underline{c1}$. c2 is the number of steps necessary to step from the present value of x to $\underline{x_f}$ using the current step size h . Initially, c2 = c1; as each step is taken c2 is decremented by one and the present value of x is computed by $x = \underline{x_f} - h \cdot \underline{c2}$. If h is halved, c1 and c2 are doubled; if h is doubled, c1 and c2 are halved. Hence c2 need not be integral.

The error vectors $\vec{e_a}$ and $\vec{e_r}$, like \vec{y} , have n components. (Although the base of the arrays y, yl, ea, and er is zero, the n components are placed

in positions 1, 2, . . . , \underline{n} of the arrays.) The procedure's error control attempts to guarantee that, in integrating from \underline{x}_i to \underline{x}_f , each component of \vec{y} will not be in absolute error more than the corresponding component of \vec{ea} and will not be in relative error more than the corresponding component of \vec{er} . At each step, the procedure requires that for each i , $1 \leq i \leq \underline{n}$, either the absolute error in $\underline{y} [i]$ does not exceed $\underline{ea} [i]/(\underline{cl}^{\underline{p}})$ or the relative error in $\underline{y} [i]$ does not exceed $\underline{er} [i]/(\underline{cl}^{\underline{p}})$.

If $\underline{p} = 1$ and $\vec{er} = 0$ then the accumulated error in any component of \vec{y} cannot exceed the corresponding component of \vec{ea} . If the error is assumed to accumulate randomly as the square root of the number of steps, $\underline{p} = 1/2$ and $\vec{er} = 0$ will cause the accumulated error in any component of \vec{y} to be approximately the corresponding component of \vec{ea} .

If $\underline{p} = 1$ and $\vec{ea} = 0$ then the accumulated error in any component of \vec{y} cannot exceed the corresponding component of \vec{er} times the largest value assumed by that component of \vec{y} during the integration. If the error is assumed to accumulate randomly as the square root of the number of steps, $\underline{p} = 1/2$ and $\vec{ea} = 0$ will cause the accumulated error in any component of \vec{y} to be approximately the corresponding component of \vec{er} times some average value assumed by that component of \vec{y} during the integration.

The procedure \underline{f} which computes $\vec{f} = \vec{f}(x, \vec{y})$ has the following declaration:

```
procedure f (n, x, xl, yv, yvl, fv, fvl);
```

```
value n;
```

```
integer n;
```

```
real x, xl;
```

```
real array yv, yvl, fv, fvl [0];
```

The parameters of the procedure f are defined as follows:

n - the number of dependent variables in the vectors \vec{y} and \vec{F}

x - the value of the independent variable

yv, vy1 - the arrays in which the high and low halves, respectively, of \vec{y} are stored

fv, fv1 - the array in which the high and low halves, respectively, of \vec{F} are stored after computation.

The procedure start is the double-precision general multistep method starting procedure described in paragraph E of this chapter. The procedure dshanks is the double-precision Runge-Kutta-Shanks integration procedure described in paragraph D of this chapter. The coefficient arrays rkscoeff and rkscoeffl contain the high and low halves, respectively, of the Runge-Kutta-Shanks coefficients in the order required by the procedure start and dshanks. The number of function evaluations rksfn is required by both start and dshanks; the order rksorder is required by dshanks.

The fudge factor boogerfactor, called δ for the remainder of this section, is necessary to make the difference between the predicted and corrected values of \vec{y}_n a better estimate of the error at each step. The computed estimate, before division by δ , is far larger than the actual step error.

The array butchercoeff contains the high halves of the coefficients of (1-2), (1-3), and (1-4) in the order $A_1, B_1, a_1, b_1, \alpha_1, \beta_1, \dots, A_k, B_k, a_k, b_k, \alpha_k, \beta_k$. A_1 is in the zero position of the array. The array butchercoeffl contains the low halves in the same order.

The suggested initial step size dx is optional. The procedure first sets cl = 2 and doubles cl until $\text{cl} \geq k-1$. If dx = 0 or dx \neq 0 and $h \leq |\text{dx}|$ then cl is

left alone. Otherwise, c1 is doubled until $h \leq |\underline{dx}|$.

The integration now begins.

$$\vec{f}_0 = \vec{f}(x_0, \vec{y}_0)$$

is computed. The start procedure is called to obtain

$$\{\vec{f}_i\}_{i=1}^{k-1}, \quad \{\vec{y}_i\}_{i=1}^{k-1}, \quad \text{and } x_{k-1}.$$

c1 and c2 are adjusted if h was changed by the start procedure. c2 is decremented by $k-1$ since $k-1$ steps took place in the start procedure. If $\underline{c2} < 1$, closing takes place. Otherwise n is set equal to k . Then the following sequence takes place.

(1-2) is used to compute $\vec{y}_{n-\theta}$; from this $\vec{f}_{n-\theta}$ is computed. Then (1-3) is used to predict \vec{y}_n ; from this \vec{f}_n is computed. Finally, (1-4) is used to correct \vec{y}_n . Let \vec{v} be the vector which is the difference between the predicted value of \vec{y}_n and the corrected value of \vec{y}_n ; \vec{v} is used as a measure of the accuracy of the step.

First, each component of \vec{v} is compared with the corresponding component of $\delta \cdot \vec{ea} / (\underline{c1}^p \cdot 2^{2k+4})$ for absolute error and with the corresponding component of $\delta \cdot \vec{er} / (\underline{c1}^p \cdot 2^{2k+4})$ times the corresponding component of the corrected value of \vec{y}_n for relative error. If no component of \vec{v} exceeds either the absolute or the relative error test, the present step size is considered too small and doubling is called for. If some component of \vec{v} exceeds either the absolute or the relative error test, this component and all remaining components are compared with the corresponding component of $\delta \cdot \vec{ea} / \underline{c1}^p$ for absolute error

and with the corresponding component of $\delta \cdot \vec{e}_r / \underline{c1}^p$ times the corresponding component of the corrected value of \vec{y}_n for relative error. If no component of \vec{v} exceeds either this absolute or this relative error test, the present step size is considered adequate. If some component of \vec{v} exceeds either this absolute or this relative error test, the present step size is considered too large and halving is called for.

If doubling is called for during three consecutive steps and if sufficient history ($2k-1$ points) is available with the present step size, the step is considered accepted, $\underline{c2}$ is decremented, the corrected value of \vec{y}_n is used to compute $\vec{f}_n = \vec{f}(x_n, \vec{y}_n)$, $\underline{c1}$ and $\underline{c2}$ are halved, and the step size h is doubled. If $\underline{c2} < 1$, closing takes place; otherwise

$$\{\vec{y}_i\}_{i=0}^{k-1}$$

becomes

$$\{\vec{y}_{n-2k+2+2i}\}_{i=0}^{k-1},$$

$$\{\vec{f}_i\}_{i=0}^{k-1}$$

becomes

$$\{\vec{f}_{n-2k+2+2i}\}_{i=0}^{k-1},$$

x_{k-1} becomes x_n , and n is set equal to k . Then control returns to the point at which (1-2) is used to compute $\vec{y}_{n-\theta}$

If the present step size is considered adequate, or if doubling is called for without being called for during three consecutive steps or with insufficient history available, the step is considered accepted, $\underline{c2}$ is decremented, and the corrected value of \vec{y}_n is used to compute $\vec{f}_n = \vec{f}(x_n, \vec{y}_n)$. If $\underline{c2} < 1$ closing takes place; otherwise n is set equal to $n+1$ and control returns to the point at which (1-2) is used to compute $\vec{y}_{n-\theta}$

If halving is called for, the step is rejected, $\underline{c1}$ and $\underline{c2}$ are doubled, and the step size h is halved. If $\underline{c2} < k-1$, closing takes place; otherwise x_θ becomes x_{n-1} , \vec{y}_θ becomes \vec{y}_{n-1} , \vec{f}_θ becomes \vec{f}_{n-1} , and control returns to the point at which the start procedure is called.

Closing takes place whenever the next step using the formulas (1-2), (1-3) and (1-4) or the next $k-1$ steps using the start procedure would carry the integration beyond \underline{xf} . If $\underline{c2} > 0$, the Runge-Kutta-Shanks procedure is used to integrate from the present value of x to \underline{xf} ; if $\underline{c2} = 0$, the integration is already complete.

Several efficiency measures are employed in the program. First, the coefficients $\{B_j\}_{j=1}^k$, $\{b_j\}_{j=1}^k$, $\{\beta_j\}_{j=0}^k$, b , and β are multiplied by the step size h and stored as multiplied until the step size changes. Second, the vectors $\delta \cdot \vec{ea} / (\underline{c1}^p \cdot 2^{2k+4})$, $\delta \cdot \vec{er} / (\underline{c1}^p \cdot 2^{2k+4})$, $\delta \cdot \vec{ea} / \underline{c1}^p$, and $\delta \cdot \vec{er} / \underline{c1}^p$ are calculated from δ , \vec{ea} , and \vec{er} and stored as calculated until the step size (and $\underline{c1}$) changes. Third, cyclic indexing is used to avoid moving the \vec{y} value and function value histories after each step or set of steps unless doubling takes place.

3. Flow Diagram and Program Listing

Figure 2 is the flow diagram for Butcher's method. The program listing follows at the end of this section.

4. Results and Conclusions

Butcher's method requires three function evaluations per accepted step; this would seem to make it inferior to Adams' or Cowell's method, either of which often runs for large numbers of steps using two function evaluations

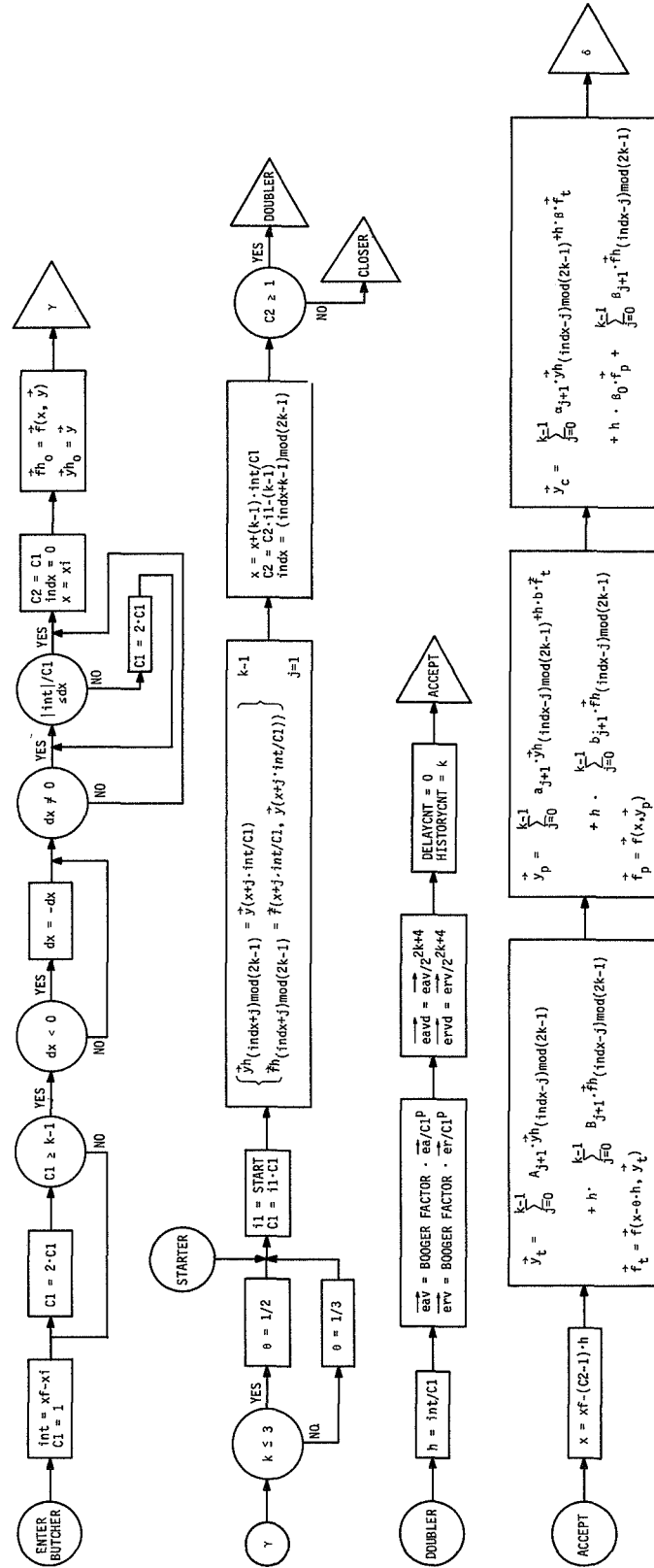


Figure 2. Flow Diagram for the Stetter-Gragg-Butcher Method.

(one for the predictor and one for the corrector) per accepted step.

However, the step size required by Butcher's method is usually considerably larger than that required by Adams' or Cowell's method for the same accuracy; hence Butcher's method takes less steps, and the number of function evaluations required by all three multistep methods is approximately the same for a given accuracy.

Butcher's method gains in order of accuracy for a given number of history points on which the next computed value is based. The price paid, however, is in the need to use \vec{y} history as well as \vec{f} history. The total number of terms in each of the sums (1-2), (1-3), and (1-4) is comparable to those used in the other multistep methods for the same order of accuracy. From this standpoint, it is not clear why Butcher's method takes larger steps than the other multistep methods.

The corrector (1-4) can only be applied once. The other multistep correctors can be repeatedly applied for some gain in accuracy without step-size change. From this point of view, perhaps it is not unusual that Butcher's method takes larger steps than the other multistep methods for a given accuracy. A method which is designed to use the corrector only once should produce the same accuracy as a method which allows repeated corrector iteration; however, the need of repeated corrector iteration usually indicates the need of a smaller step size.

The step-size control employed in this program seems also to reflect the fact that the corrector cannot be iterated. The difference between predicted and corrected values is used as an estimate of error at each step; this has been used with considerable success in Adams' method. However,

results obtained from the restricted three-body orbits show that this measure is not an entirely reliable one for Butcher's method. No one fudge factor δ was found to be satisfactory for all values of k ; moreover, for a given k , no value of δ was found to be satisfactory for all accuracies. For each order the optimum value of δ increased as higher accuracies were asked; for each accuracy the optimum value of δ was somewhat higher for larger values of $k \geq 2$. For $k=1$, the program's step size control was entirely unsatisfactory, this value of k produces a third order Runge-Kutta type formula, where the predicted value is like one of the intermediate calculations and which does not bear too much resemblance to the final computed value.

The decaying exponential gave interesting results with Butcher's method. The number of function evaluations was a linear function of error obtained right up to the limits of the machine's accuracy. There was no splitting into branches, as shown by the corresponding results obtained by Cowell's method; this could well be due to the fact that Butcher's method allows only one corrector application and requires a decrease in step size if one application is not sufficient.

```

PROCEDURE BUTCHER(N,XI,XFL,XF,XIL,Y,YL,F,EA,ER,P,DX,RKSFN,RKSORDER,00440000
,RKSCOEFF,RKSCOEFFL,BODGERFACTOR,K,BUTCHERCOEFF,BUTCHERCOEFFL,
START,DSHANKS);
00441000
00442000
VALUE N,XI,XIL,XF,XFL,P,DX,RKSFN,RKSORDER,K ;
00443000
INTEGER N,RKSFN,RKSORDER,K ;
00444000
REAL XI,XIL,BODGERFACTOR,XF,XFL,P,DX ;
00445000
REAL ARRAY Y,YL,EA,ER,RKSCOEFF,RKSCOEFFL,BUTCHERCOEFF,
BUTCHERCOEFFL(0);
00446000
00447000
PROCEDURE F,DSHANKS ;
00448000
INTEGER PROCEDURE START ;
00449000
00450000
00451000
00452000
00453000
00454000
00455000
00456000
00457000
00458000
00459000
00460000
00461000
00462000
00463000
00464000
00465000
00466000
00467000
00468000
00469000
00470000
00471000
00472000
00473000
00474000
00475000
00476000
00477000
00478000

BEGIN
INTEGER I,J,C1,KM1,TKM1,INDX,I1,I2,CYI,DELAYCNT,HISTORYCNT,TKM2
;
REAL INT,INTL,C2,C2L,DFACTOR,X,XL,H,HL,T1,T1L,T2,T2L,T3,T3L,T4,
T4L,T5,T5L,T6,T6L,T7,T7L,T8,T8L,THETA,THETAL,THETAHL,HB,
HBL,HBETA,HBETAL,HBETA0,HBETAOL,XI,XIL ;
REAL ARRAY YH,YHL,FH,FHL(0:K +K -2,0:N),YT,YTL,YP,YPL,YC,YCL,FT
,FTL,EAV,ERV,EAVD,ERVD(0:N),TCOEFF,TCOEFFL,PCOEFF,PCOEFFL,CCOEFF
,CCOEFFL,HTCOEFF,HTCOEFFL,HPCOEFF,HPCOEFFL,HCCOEFF,HCCOEFFL(0:K
-1);
LABEL LO,L1,L2,L3,L4,ACCEPT,DOUBLER,STARTER,CLOSER ;
DOUBLE(XF,XFL,XI,XIL,=,INT,INTL);
C1 :=1 ;
KM1 :=K -1 ;
LO:C1 :=C1 +C1 ;
IF C1 <KM1 THEN GO TO LO ;
IF DX <0 THEN DX :=-DX ;
IF DX #0 THEN
BEGIN
L1:IF ABS(INT)/C1 >DX THEN
BEGIN
C1 :=C1 +C1 ;
GO TO L1
END
END ;
DOUBLE(C1,0,1,0,x,=,C2,C2L);
I1 :=-2 ;
FOR I :=0 STEP 1 UNTIL KM1 DO

```

```

BEGIN
TCOEFF[I]:=BUTHERCOEFF[I1:=I1+2];
TCOEFF[I1:=BUTHERCOEFF[I1];
PCOEFF[I1:=BUTHERCOEFF[I1:=I1+2];
PCOEFF[I1:=BUTHERCOEFF[I1];
CCOEFF[I1:=BUTHERCOEFF[I1:=I1+2];
CCOEFF[I1:=BUTHERCOEFF[I1];
END;
TKM1:=TKM1+K;
TKM2:=TKM1-1;
DFACTOR:=2.0*(2*K+4);
FOR J:=1 STEP 1 UNTIL N DO
BEGIN
YH[0,J]:=Y[J];
YH[0,J]:=YH[J]
END;
INDX:=0;
X:=XI;
XL:=XIL;
IF K<3 THEN DOUBLE(0.5,THETA,THETA,ELSE DOUBLE(1.0,3.0,
,THETA,THETA);
F(N,X,XL,Y,YL,FH[0,],FHL[0,]);
STARTER:C1:=(I1:=START(N,XI,XIL,XF,XFL,C1,EA,ER,F,KM1,X,XL,YT
,YTL,YH,YHL,FH,FHL,YT,YTL,INDX,TKM1,2,P,RKSFN,RKSCOEFF,RKSCOEFFL
))X C1;
DOUBLE(C2,C2L,I1,0,X,KM1,0,0,0,0,C2,C2L);
INDX:=(INDX+KM1)MOD TKM1;
IF C2<1 THEN GO TO CLOSER;
DOUBLER:DOUBLE(INT,INTL,C1,0,0,0,0,H,HL);
I1:=1;
FOR I:=0 STEP 1 UNTIL KM1 DO
BEGIN
DOUBLE(H,HL,BUTHERCOEFF[I1],BUTHERCOEFF[I1],X,0,HTCOEFF[I1],
HTCOEFF[I1]);
I1:=I1+2;
DOUBLE(H,HL,BUTHERCOEFF[I1],BUTHERCOEFF[I1],X,0,HPCOEFF[I1],
HPCOEFF[I1]);
I1:=I1+2;
DOUBLE(H,HL,BUTHERCOEFF[I1],BUTHERCOEFF[I1],X,0,HCCOEFF[I1],
HCCOEFF[I1]);

```

```

I1 := I1 + 2
END ;
DOUBLE(H,HL,BUTCHERCOEFF[I1],BUTCHERCOEFFL[I1],X, :=,HBETA,HBETAL,
);
I1 := I1 - 1 ;
DOUBLE(H,HL,BUTCHERCOEFF[I1],BUTCHERCOEFFL[I1],X, :=,HB, HBL);
I1 := I1 + 2 ;
DOUBLE(H,HL,BUTCHERCOEFF[I1],BUTCHERCOEFFL[I1],X, :=,HBETA0,
HBETA0L);
T1 := (C1 * P) / BOOGERFACTOR ;
FOR J := 1 STEP 1 UNTIL N DO
BEGIN
    EAVD[J] := (EAV[J] := EA[J] / T1) / DFACTOR ;
    ERVD[J] := (ERV[J] := ER[J] / T1) / DFACTOR
END ;
DOUBLE(H,HL,THETA,THETAL,X, :=,THETA,THETAHL);
DELAYCNT := 0 ;
HISTORYCNT := K ;
ACCEPT := DOUBLE(XF,XFL,C2,C2L,1.0, -,H,HL,X, :=,X,XL);
DOUBLE(X,XL,THETA,THETAHL, :=,XT,XTL);
T1 := TCUEFFL[0];
T1L := TCUEFFL[0];
T2 := HTCOEFFL[0];
T2L := HTCOEFFL[0];
T3 := PCUEFFL[0];
T3L := PCUEFFL[0];
T4 := HPCUEFFL[0];
T4L := HPCUEFFL[0];
T5 := CCUEFFL[0];
T5L := CCUEFFL[0];
T6 := HCCUEFFL[0];
T6L := HCCUEFFL[0];
FOR J := 1 STEP 1 UNTIL N DO
BEGIN
    T7 := YH[INDX,J];
    T7L := YHL[INDX,J];
    T8 := FH[INDX,J];
    T8L := FHL[INDX,J];
    DOUBLE(T7,T7L,T1,T1L,X,T8,T8L,T2,T2L,X, :=,YT[J],YTL[J]);
    DOUBLE(T7,T7L,T3,T3L,X,T8,T8L,T4,T4L,X, :=,YP[J],YPL[J]);

```

00519000

00520000

00521000

00522000

00523000

00524000

00525000

00526000

00527000

00528000

00529000

00530000

00531000

00532000

00533000

00534000

00535000

00536000

00537000

00538000

00539000

00540000

00541000

00542000

00543000

00544000

00545000

00546000

00547000

00548000

00549000

00550000

00551000

00552000

00553000

00554000

00555000

00556000

00557000

00558000

```

DOUBLE(T7,T7L,T5,T5L,X,T8,T8L,T6,T6L,X,+,YCL[J])
00559000
END;
00560000
CYI := INDX + TKM1 ;
00561000
FOR I := 1 STEP 1 UNTIL KM1 DO
00562000
BEGIN
00563000
  T1 := TC0EFF[I];
00564000
  T1L := TC0EFFL[I];
00565000
  T2 := HT0EFF[I];
00566000
  T2L := HT0EFFL[I];
00567000
  T3 := PC0EFF[I];
00568000
  T3L := PC0EFFL[I];
00569000
  T4 := HP0EFF[I];
00570000
  T4L := HP0EFFL[I];
00571000
  T5 := CC0EFF[I];
00572000
  T5L := CC0EFFL[I];
00573000
  T6 := HC0EFF[I];
00574000
  T6L := HC0EFFL[I];
00575000
  T6L := HC0EFFL[I];
00576000
  I1 := (CYI - 1) MOD TKM1 ;
00577000
  FOR J := 1 STEP 1 UNTIL N DO
00578000
  BEGIN
00579000
    T7 := YH[I1,J];
00580000
    T7L := YHL[I1,J];
00581000
    T8 := FH[I1,J];
00582000
    T8L := FHL[I1,J];
00583000
    DOUBLE(T7,T7L,T1,T1L,X,T8,T8L,T2,T2L,X,+,YT[J],YT[J],+,+);
00584000
    YT[J],YT[J];
00585000
    DOUBLE(T7,T7L,T3,T3L,X,T8,T8L,T4,T4L,X,+,YP[J],YPL[J],+,+);
00586000
    YP[J],YPL[J];
00587000
    DOUBLE(T7,T7L,T5,T5L,X,T8,T8L,T6,T6L,X,+,YC[J],YCL[J],+,+);
00588000
    YC[J],YCL[J];
00589000
  END
00590000
END ;
00591000
FCN,X,T,XTL,YT,YTL,FT,FTL);
00592000
FOR J := 1 STEP 1 UNTIL N DO
00593000
BEGIN
00594000
  T1 := FT[J];
00595000
  T1L := FTL[J];
00596000
  DOUBLE(CH8,HBL,T1,T1L,X,YP[J],YPL[J],+,+);
00597000
  DOUBLE(CHBETA,HBETAL,T1,T1L,X,YC[J],YCL[J],+,+);
00598000
END ;

```

```

F(N,X,XL,YP,YPL,FT,FTL);
INDX :=(INDX +1)MOD TKM1 ;
FOR J :=1 STEP 1 UNTIL N DO
BEGIN
DOUBLE(HBETA0,HBETA0L,FT[J],FTL[J],X,YC[J],YCL[J],+,:=,T3,T3L)
;
DOUBLE(T3,T3L,YP[J],YPL[J],+,:=,T2,T2L);
IF (T2 :=ABS(T2))>EAVD[J]THEN
BEGIN
IF T2 >ERV[J]*ABS(T3)THEN
BEGIN
IF T2 >EAV[J]THEN
BEGIN
IF T2 >ERV[J]*ABS(T3)THEN GO TO L2
END ;
YHL[INDX,J] :=T3 ;
YHL[INDX,J] :=T3L ;
FOR J :=J +1 STEP 1 UNTIL N DO
BEGIN
DOUBLE(HBETA0,HBETA0L,FT[J],FTL[J],X,YC[J],YCL[J],+,:=,
T3,T3L);
DOUBLE(T3,T3L,YP[J],YPL[J],+,:=,T2,T2L);
IF (T2 :=ABS(T2))>EAV[J]THEN
BEGIN
IF T2 >ERV[J]*ABS(T3)THEN
BEGIN
L2:C1 :=C1 +C1 ;
DOUBLE(XF,XFL,C2,C2L,H,HL,X,+,:=,X,XL);
DOUBLE(C2,C2L,C2,C2L,+,:=,C2,C2L);
INDX :=(INDX +TKM2)MOD TKM1 ;
IF C2 < KM1 THEN GO TO CLOSER ;
GO TO STARTER
END
END ;
YHL[INDX,J] :=T3 ;
YHL[INDX,J] :=T3L
END ;
DELAYCNT :=0 ;
F(N,X,XL,YHL[INDX,*],YHL[INDX,*],FHL[INDX,*]);
GO TO L3

```

```

00639000
00640000
00641000
00642000
00643000
00644000
00645000
00646000
00647000
00648000
00649000
00650000
00651000
00652000
00653000
00654000
00655000
00656000
00657000
00658000
00659000
00660000
00661000
00662000
00663000
00664000
00665000
00666000
00667000
00668000
00669000
00670000
00671000
00672000
00673000
00674000
00675000
00676000
00677000
00678000

END
END ;
YH[INDX,J]:=T3 ;
YHL[INDX,J]:=T3L
END ;
F(N,X,XL,YH[INDX,*J],YHL[INDX,*J],FH[INDX,*J],FHL[INDX,*J]);
IF DELAYCNT ≥ 2 THEN
BEGIN
IF HISTORYCNT ≥ TKM1 THEN
BEGIN
DOUBLE(C2,C2L,1.0,2.0,2.0,2.0,C2,C2L);
IF C2 < 1 THEN GO TO CLOSER ;
C1 := C1 DIV 2 ;
CYI := INDEX + TKM1 ;
FOR I := 1 STEP 1 UNTIL KM1 DO
BEGIN
I1 := (I2 := CYI - I) MOD TKM1 ;
I2 := (I2 - I) MOD TKM1 ;
FOR J := 1 STEP 1 UNTIL N DO
BEGIN
YH[I1,J] := YH[I2,J];
YHL[I1,J] := YHL[I2,J];
FH[I1,J] := FH[I2,J];
FHL[I1,J] := FHL[I2,J]
END
END ;
GO TO DOUBLER
END
END ;
DELAYCNT := DELAYCNT + 1 ;
L3: DOUBLE(C2,C2L,1.0,2.0,2.0,2.0,C2,C2L);
IF C2 ≥ 1 THEN
BEGIN
HISTORYCNT := HISTORYCNT + 1 ;
GO TO ACCEPT
END ;
CLOSER: IF C2 > 0 THEN DSHANKS(N,X,XL,XF,XFL,YH[INDX,*J],YHL[INDX,*J],F,RKSFN,RKSORDER,RKSCOEFF,RKSCOEFFL,P,EA,ER,ABS(INT)/C1);
FOR J := 1 STEP 1 UNTIL N DO
BEGIN

```

```
Y[J]:=YH[INDX,J];  
YL[J]:=YH[INDX,J]  
END ;  
END ;
```

```
00679000  
00680000  
00681000  
00682000  
00683000
```

C. The Cowell Method

1. Description of the Method

Cowell's method as described herein is a multistep predictor-corrector method for the numerical solution of the first-order vector differential equation

$$\vec{y}'(x) = \frac{d}{dx} \vec{y}(x) = \vec{f}(x, \vec{y}(x)), \quad \vec{y}(x_0) = \vec{y}_0. \quad (1-1)$$

A completion derivation and description of Cowell's method can be found in [7] and [9]; only the essential formulas are included here.

The following notation is adopted. Let q be an even positive integer, $m = q/2$, h be the step size (assumed to be constant over some set of calculations),

$$x_n = x_0 + nh, \quad \vec{y}_n = \vec{y}(x_n), \text{ and } \vec{f}_n = \vec{f}(x_n, \vec{y}_n).$$

The predictor formula is

$$\vec{y}_n = h [\delta^{-1} \vec{f}_{n-1/2} + \sum_{j=0}^q P_j \vec{f}_{n-1-j}], \quad (1-2)$$

The corrector formula is

$$\vec{y}_n = h [\delta^{-1} \vec{f}_{n-1/2} + \sum_{j=0}^q C_j \vec{f}_{n-j}], \quad (1-3)$$

and the mid-range formula is

$$\vec{y}_n = h [\delta^{-1} \vec{f}_{n-1/2} + \sum_{j=0}^q M_j \vec{f}_{n+m-j}]. \quad (1-4)$$

The predictor formula gives \vec{y}_n in terms of $\delta^{-1} \vec{f}_{n-1/2}$ and the function values

at the previous $q+1$ points; the corrector formula gives a new value of \vec{y}_n in terms of $\delta^{-1}\vec{f}_{n-1/2}$, the old value of \vec{y}_n , and the function values at the previous q points; the mid-range formula gives a value of \vec{y}_n in terms of $\delta^{-1}\vec{f}_{n-1/2}$ and the function values at the $q+1$ consecutive points centered around x_n .

The equation

$$\delta^{-1}\vec{f}_{n-1/2} = \delta^{-1}\vec{f}_{n-1-1/2} + \vec{f}_{n-1} \quad (1-5)$$

completes the set of formulas necessary for the numerical solution of (1-1).

If it is assumed that

$$\{\vec{f}_i\}_{i=0}^q \quad \text{and} \quad \vec{y}_m$$

have been obtained by some starting procedure, the mid-range formula (1-4) can be applied with $n = m$ to obtain

$$\delta^{-1}\vec{f}_{m-1/2}.$$

Equation (1-5) can then be applied m times to obtain

$$\delta^{-1}\vec{f}_{q-1/2}.$$

For each positive integer i

$$\delta^{-1}\vec{f}_{q+i-1/2}$$

can be computed from

$$\delta^{-1}\vec{f}_{q+i-1-1/2}$$

and \vec{f}_{q+i-1} using (1-5); \vec{y}_{q+i} can be computed using the predictor (1-2); \vec{f}_{q+i} can be computed from the predicted value; \vec{y}_{q+i} can be computed using the corrector (1-3); \vec{f}_{q+i} can be computed from the corrected value; if necessary, iteration can be resorted to, using (1-3), until the last two computed values of \vec{y}_{q+i} agree to sufficient accuracy. For any $j \geq m$ a value of \vec{y}_{q+j-m} can be obtained from the mid-range formula (1-4) and compared with the value obtained from the predictor-corrector step. If the two values of \vec{y}_{q+j-m} are in sufficient agreement, the values up through \vec{y}_{q+j} are considered acceptable; if not, \vec{y}_{q+j-m} is considered the last acceptable value and all values beyond are rejected.

Hence, the knowledge of (1-2), (1-3), (1-4), and (1-5) is sufficient to apply Cowell's method in the numerical solution of (1-1). The coefficients $\{P_j\}_{j=0}^q$, $\{C_j\}_{j=0}^q$, and $\{M_j\}_{j=0}^q$ are given in [] for $q = 4, 6, 8, 10, 12, 14$, and 16 .

2. The Computer Program

The Cowell computer program is a Burroughs B-5500 ALGOL double-precision procedure whose declaration is as follows:

```

procedure cowell (n, xi, xil, xf, xfl, y, yl, f, ea, er, p, dx, rksfn,
                  rksorder, rkscoeff, rkscoeffl, q, cowellcoeff,
                  cowellcoeffl, start, dshanks);
value n, xi, xil, xf, xfl, p, dx, rksfn, rksorder, q;
integer n, rksfn, rksorder, q;
real xi, xil, xf, xfl, p, dx;
real array y, yl, ea, er, rkscoeff, rkscoeffl, cowellcoeff, cowellcoeffl [0];
procedure f, dshanks;
integer procedure start;
```

The parameters of the procedure are defined as follows:

n - the number of dependent variables in the vectors \vec{y} and \vec{f}

xi, xil - the high and low halves, respectively, of x_0 , the starting value of the independent variable x

xf, xfl - the high and low halves, respectively, of the final value of the independent variable x

y, yl - the arrays in which are located the high and low halves, respectively, of $\vec{y}_0 = \vec{y}(\underline{xi}, \underline{xil})$ upon entry and of $\vec{y}(\underline{xf}, \underline{xfl})$ upon exit

f - the double procedure which computes $\vec{f} = \vec{f}(x, \vec{y})$

ea - the array containing the absolute error vector

er - the array containing the relative error vector

p - the exponent used in step-size control

dx - the suggested initial step size

rksfn - the number of function evaluations used in the Runge-Kutta-Shanks starting and closing procedures

rksorder - the order of the Runge-Kutta-Shanks closing procedure

rkscoeff, rkscoeffl - the arrays containing the high and low halves, respectively, of the Runge-Kutta-Shanks coefficients for the starting and closing procedures

q - the even integer used in describing Cowell's method

cowellcoeff, cowellcoeffl - the arrays containing the high and low halves, respectively, of the Cowell coefficients

start - the double-precision starting procedure

dshanks - the double-precision closing procedure

The procedure performs the numerical integration of (1-1) in double precision from $x = \underline{x_i}$ to $x = \underline{x_f}$. (For convenience of description, the low half of a variable is often not mentioned.) The step size h used is always the length of the interval $\underline{x_f} - \underline{x_i}$ divided by a power of 2. In order to avoid error build-up in the independent variable two counters, $\underline{c_1}$ and $\underline{c_2}$, are kept. $\underline{c_1}$ is always a positive, integral power of 2, and $h = (\underline{x_f} - \underline{x_i})/\underline{c_1}$. $\underline{c_2}$ is the number of steps necessary to step from the present value of x to $\underline{x_f}$ using the current step size h . Initially, $\underline{c_2} = \underline{c_1}$; as each step is taken $\underline{c_2}$ is decremented by one and the present value of x is computed by $x = \underline{x_f} - h \cdot \underline{c_2}$. If h is halved, $\underline{c_1}$ and $\underline{c_2}$ are doubled; if h is doubled, $\underline{c_1}$ and $\underline{c_2}$ are halved. Hence $\underline{c_2}$ need not be integral.

The error vectors $\vec{e_a}$ and $\vec{e_r}$, like \vec{y} , have \underline{n} components. (Although the base of the arrays \underline{y} , $\underline{y_1}$, $\underline{e_a}$, and $\underline{e_r}$ is zero, the \underline{n} components are placed in positions 1, 2, . . . , \underline{n} of the arrays.) The procedure's error control attempts to guarantee that, in integrating from $\underline{x_i}$ to $\underline{x_f}$, each component of \vec{y} will not be in absolute error more than the corresponding component of $\vec{e_a}$ and will not be in relative error more than the corresponding component of $\vec{e_r}$. At each step, the procedure requires that for each i , $1 \leq i \leq \underline{n}$, either the absolute error in $\underline{y} [i]$ does not exceed $\underline{e_a} [i]/(\underline{c_1}^{\underline{p}})$ or the relative error in $\underline{y} [i]$ does not exceed $\underline{e_r} [i]/(\underline{c_1}^{\underline{p}})$.

If $\underline{p} = 1$ and $\vec{e_r} = 0$ then the accumulated error in any component of \vec{y} cannot exceed the corresponding component of $\vec{e_a}$. If the error is assumed to accumulate randomly as the square root of the number of steps, $\underline{p} = 1/2$ and $\vec{e_r} = 0$ will cause the accumulated error in any component of \vec{y} to be approximately the corresponding component of $\vec{e_a}$.

If $\underline{p} = 1$ and $\vec{ea} = 0$ then the accumulated error in any component of \vec{y} cannot exceed the corresponding component of \vec{er} times the largest value assumed by that component of \vec{y} during the integration. If the error is assumed to accumulate randomly as the square root of the number of steps, $\underline{p} = 1/2$ and $\vec{ea} = 0$ will cause the accumulated error in any component of \vec{y} to be approximately the corresponding component of \vec{er} times some average value assumed by that component of \vec{y} during the integration.

The procedure f which computes $\vec{f} = \vec{f}(x, \vec{y})$ has the following declaration:

```
procedure f (n, x, xl, yv, yvl, fv, fvl);
```

```
value n;
```

```
integer n;
```

```
real x, xl;
```

```
real array yv, yvl, fv, fvl [0];
```

The parameters of the procedure f are defined as follows:

n - the number of dependent variables in the vectors \vec{y} and \vec{f}

x - the value of the independent variable

yv, yvl - the arrays in which the high and low halves, respectively, of \vec{y} are stored

fv, fvl - the array in which the high and low halves, respectively, of \vec{f} are stored after computation.

The procedure start is the double-precision general multistep method starting procedure described in paragraph E of this chapter. The procedure dshanks is the double-precision Runge-Kutta-Shanks integration procedure described in paragraph D of this chapter. The coefficient arrays rkscoeff and rkscoeffl contain the high and low halves, respectively, of the Runge-

Kutta-Shanks coefficients in the order required by the procedures start and dshanks. The number of function evaluations rksfn is required by both start and dshanks; the order rksorder is required by dshanks.

The array cowellcoeff contains the high halves of the coefficients of (1-2), (1-3), and (1-4) in the order $P_0, P_1, \dots, P_q, C_0, C_1, \dots, C_q, M_0, M_1, \dots, M_q$. P_0 is in the zero position of the array. The array cowellcoeffl contains the low halves in the same order.

The suggested initial step size dx is optional. The procedure first sets c1 = 2 and doubles c1 until c1 \geq q. If dx = 0 or dx \neq 0 and $h \leq |\underline{dx}|$ then c1 is left alone. Otherwise, c1 is doubled until $h \leq |\underline{dx}|$.

The integration now begins.

$$\vec{f}_0 = \vec{f}(x_0, \vec{y}_0)$$

is computed. The start procedure is called to obtain

$$\{\vec{f}_i\}_{i=1}^q, \vec{y}_m, \vec{y}_q \text{ and } x_q.$$

c1 and c2 are adjusted if h was changed by the start procedure. c2 is decremented by q since q steps took place in the start procedure. If c2 < m, closing takes place. Otherwise,

$$\delta^{-1} \vec{f}_{m-1/2}$$

is calculated from

$$\{\vec{f}_i\}_{i=0}^q$$

and \vec{y}_m using the mid-range formula (1-4). m applications of (1-5) yield

$$\delta^{-1} \vec{f}_{q-1/2}$$

and n is set equal to q .

For $1 \leq i \leq m$ the following set of steps takes place. $c2$ is decremented by 1, and x_{n+i} is calculated.

$$\delta^{-1} \vec{f}_{n+i-1/2}$$

is calculated from

$$\delta^{-1} \vec{f}_{n+i-1-1/2}$$

and \vec{f}_{n+i-1} using (1-5). \vec{y}_{n+i} is calculated using the predictor (1-2), and \vec{f}_{n+i} is calculated. \vec{y}_{n+i} is next calculated using the corrector (1-3), and \vec{f}_{n+i} is again calculated. Let \vec{v} be the vector which is the absolute value of the difference between the last two calculated values of \vec{y}_{n+i} . Each component of \vec{v} is compared with the corresponding component of $\vec{ea}/(10 \cdot \underline{cl}^p)$ for absolute error and with the product of the corresponding components of $\vec{er}/(10 \cdot \underline{cl}^p)$ and the last calculated value of \vec{y}_{n+i} for relative error. If any component of \vec{v} exceeds in both the absolute and the relative error tests, the steps which calculate \vec{y}_{n+i} using the corrector (1-3), calculate \vec{f}_{n+i} from the value of \vec{y}_{n+i} , and which test the last two calculated values of \vec{y}_{n+i} are repeated. When each component of \vec{v} does not exceed in either the absolute or the relative error test, the last values of \vec{y}_{n+i} and \vec{f}_{n+i} are retained.

The mid-range formula (1-4) is now used to calculate a new value of \vec{y}_n from

$$\{\vec{f}_{n+i}\}_{i=-m}^m$$

and

$$\delta^{-1} \vec{f}_{n-1/2}.$$

Let \vec{v} be the vector which is the absolute value of the differences between the new value of \vec{y}_n and the previously calculated value of \vec{y}_n . If sufficient history is available for doubling the step size, i. e., $n > q + m$, each component of \vec{v} is compared with the corresponding component of $\vec{ea}/(10 \cdot \underline{cl}^p \cdot 2^{q+3})$ for absolute error and with the product of the corresponding components of $\vec{er}/(10 \cdot \underline{cl}^p \cdot 2^{q+3})$ and the new value of \vec{y}_n for relative error.

If each component of \vec{v} does not exceed in either the absolute or the relative error tests, the last m steps are accepted, \underline{cl} and $\underline{c2}$ are halved, and the step size is doubled. If $\underline{c2} < m$, closing takes place. Otherwise

$$\{\vec{f}_i\}_{i=0}^q$$

becomes

$$\{\vec{f}_{n-m+2i}\}_{i=0}^q,$$

\vec{y}_m becomes \vec{y}_{n-m} , \vec{y}_q becomes \vec{y}_{n+m} , x_q becomes x_{n+m} , and, as if the starting procedure had calculated these values, control returns to the step where

$$\delta^{-1} \vec{f}_{m-1/2}$$

is calculated using the mid-range formula (1-4).

If any component of \vec{v} exceeds in both the absolute and the relative error tests, this component and each untested component is compared with the corresponding component of $\vec{ea}/(10 \cdot \underline{cl}^p)$ for absolute error and with the product of the corresponding components of $\vec{er}/(10 \cdot \underline{cl}^p)$ and the new value of \vec{y}_n for relative error. If each component of \vec{v} does not exceed in either the absolute or the relative error test, the last m steps are accepted and the step size remains unchanged. If $\underline{c2} < m$, closing takes place. Otherwise, n becomes $n + m$ and control returns to the steps which calculate

$$\{\vec{y}_{n+i}\}_{i=1}^m.$$

If any component of \vec{v} exceeds in both the absolute and the relative error tests, the last m steps are rejected, $\underline{c2}$ is incremented by m , $\underline{c1}$ and $\underline{c2}$ are doubled, and the step size is halved. \vec{f}_0 becomes \vec{f}_n , \vec{y}_0 becomes \vec{y}_n , x_0 becomes x_n , and control is returned to the step which calls the start procedure.

If sufficient history is not available for doubling, control transfers as if the first component of \vec{v} exceeded both the first component of $\vec{ea}/(10 \cdot \underline{cl}^p \cdot 2^{q+3})$ and the product of the first components of $\vec{er}/(10 \cdot \underline{cl}^p \cdot 2^{q+3})$ with the first component of the new value of \vec{y}_n .

Closing takes place whenever m steps at the present step size would carry the integration beyond \underline{xf} , i. e., whenever $\underline{c2} > 0$, the Runge-Kutta-Shanks procedure is used to integrate from the present value of x to \underline{xf} ; if $\underline{c2} = 0$, the present value of x is \underline{xf} . In either case, the integration is now complete.

Several efficiency measures are employed in the program. First, the

coefficients

$$\{P_j\}_{j=0}^q,$$

$$\{C_j\}_{j=0}^q,$$

and

$$\{M_j\}_{j=0}^q$$

are multiplied by the step size h and stored as multiplied until the step size changes. Second, the vectors $\vec{ea}/(10 \cdot \underline{cl}^p)$, $\vec{er}/(10 \cdot \underline{cl}^p)$, $\vec{ea}/(10 \cdot \underline{cl}^p \cdot 2^{q+3})$, and $\vec{er}/(10 \cdot \underline{cl}^p \cdot 2^{q+3})$ are calculated from \vec{ea} and \vec{er} and stored as calculated until the step size changes. Third, the corrector partial sum

$$h\delta^{-1}\vec{f}_{n-1/2} + h\sum_{j=1}^q C_j \vec{f}_j$$

is computed and stored at each step; successive applications of the corrector only require adding $h \cdot C_0 \cdot \vec{f}_n$ to obtain \vec{y}_n . Fourth, during applications of the corrector, two arrays are used to store the last two calculated values of \vec{y}_n ; a flag is used to mark the last calculated value so that the next value is placed in the unflagged array and the flag is switched. This avoids transfer from array to array as successive corrector iterates are computed. Fifth, cyclic indexing is used to avoid moving the function value history after each step or set of steps unless doubling takes place. One unusual condition can result. If, during any step taken in computing

$$\{y_{n+i}\}_{i=1}^m,$$

the number of times through the corrector exceeds eight, control transfers as if the set of m steps has been completed and rejected, i.e., a step size halving was called for with a restart beginning at \vec{y}_n .

3. Flow Diagram and Program Listing

Figure 3 is the flow diagram for the Cowell method. The program listing follows at the end of this section.

4. Results and Conclusions

Numerous results and conclusions applying to the single-precision version of the procedure were discussed in [21]; these were again borne out in experiments performed using the double-precision version. Three significant differences might be noted, however.

First, the accuracy cutoff for the single-precision version occurred when doubling required relative error estimates of around 10^{-11} . This cutoff for the double-precision version occurred when doubling required relative error estimates of around 10^{-23} ; this increase in possible accuracy is exactly the additional accuracy afforded by double-precision.

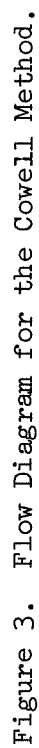
Secondly, best results in single precision seemed to occur with $q = 4$ or $q = 6$; best results in double precision seemed to occur with $q = 10$ or $q = 12$, the same as observed in [11].

Thirdly, the matching of the Runge-Kutta-Shanks starting procedure to the order q and the asked accuracy vectors \vec{e}_a and \vec{e}_r was more complex. The (5,5) Shanks formula seemed best for $q = 4, 6$ and 8 at all accuracies asked and for $q = 10$ and 12 at those accuracies obtainable in single precision. However, for $q = 10$ and 12 at most accuracies not obtainable in single precision, the (7, 9) Shanks formula seemed superior. For the

highest accuracies obtainable in double precision for $q = 10$ and 12 , the $(8, 12)$ Shanks formula seemed slightly better than the $(7, 9)$ Shanks formula. For $q = 14$ at all accuracies requiring double precision, the $(8, 12)$ formula appeared best. However, results are not entirely conclusive because of the lack of sufficiently extensive data.

Results obtained from calculations of the decaying exponential, i.e., $y' = -y$, $y(0) = 1$, show an unusual phenomenon. As increased accuracy is asked, the step size may not change. Instead, the corrector may be used two or three times instead of once, and there is a decrease in accuracy obtained. However, further increases in accuracy asked eventually produce a decrease in step size, a return to only one pass through the corrector, and an increase in accuracy obtained. This seems to indicate that a fixed number of times through the corrector, followed by step-size control based on the difference between predictor and final corrector, should be used. (The Cowell procedure uses a variable number of times through the corrector and uses a mid-range formula test for step-size control.) It should be noted that this phenomenon is not present in calculations of the restricted three-body orbits; the orbits require both doubling and halving of step size and the decaying exponential should only require halving. Moreover, the decaying exponential error control was relative; the orbit error control was absolute.

Decaying exponential results seem to show a reasonably linear relationship when function evaluations were plotted on a log-log scale as a function of error obtained. This must be qualified by two statements. First, when the limits of the accuracy of the machine



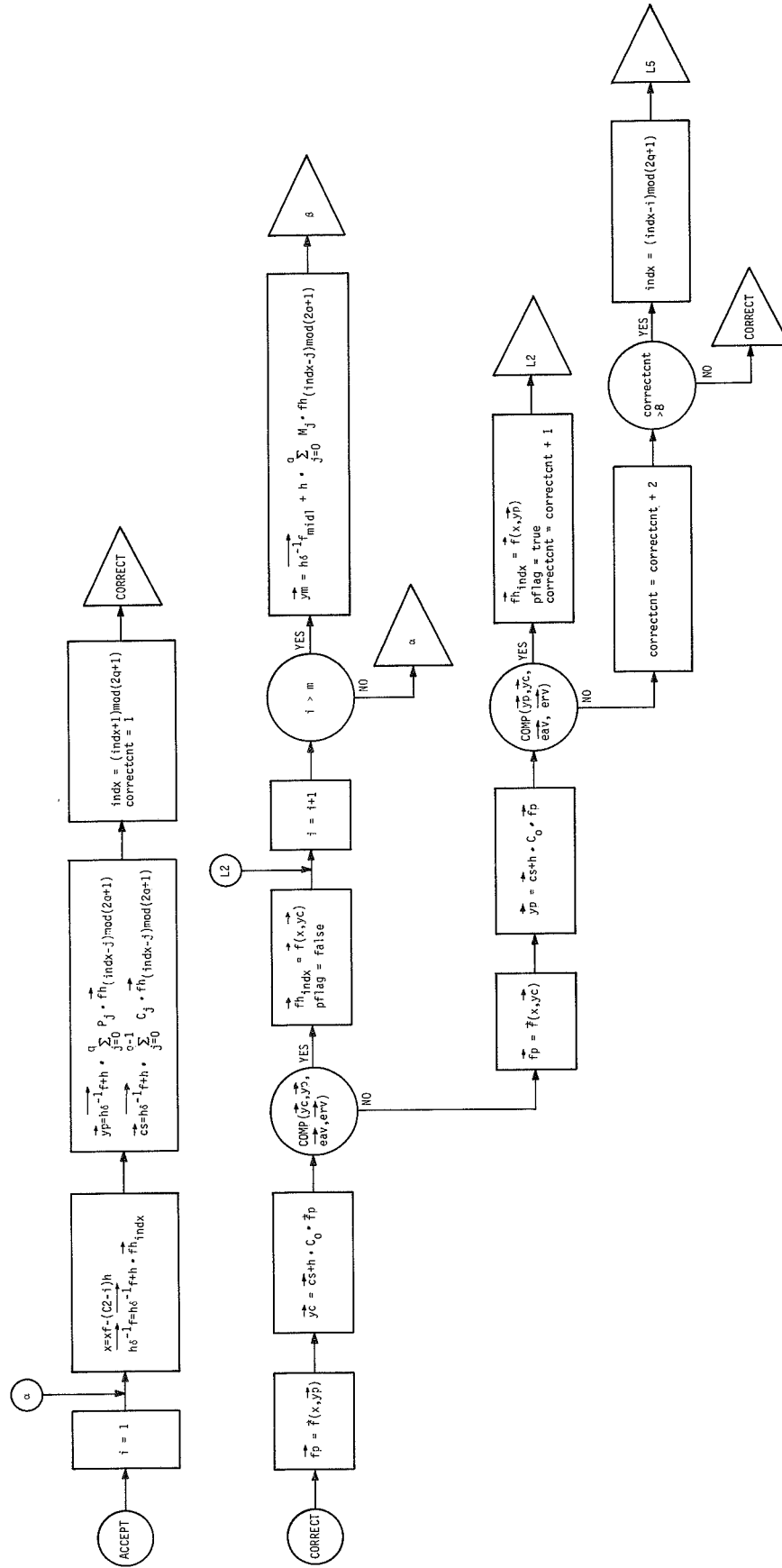
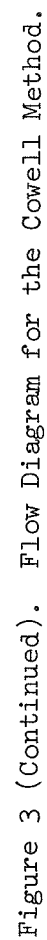


Figure 3 (Continued). Flow Diagram for the Cowell Method.



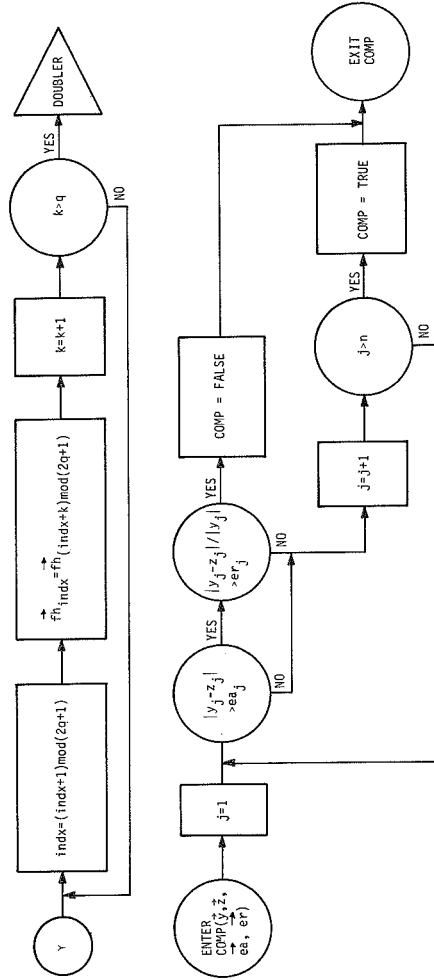


Figure 3 (Continued). Flow Diagram for the Cowell Method.

were reached, the curves turned upward vertically and gave a graphic exhibit of the limitations of the machine. Secondly, the graphs often split into two or three branches. The branches were the result of the different number of times through the corrector, and each branch was linear up to the accuracy of the machine. This linearity seems to indicate that there is little build-up of round-off error as the number of steps (and, correspondingly, the number of function evaluations) increases. However, the different branches seem to show that additional times through the corrector do not always produce the same effect as a halving of step size.

For the restricted three-body orbits, the error obtained was always as good as the error asked; for the decaying exponential, the error obtained was rarely as good as the error asked. Moreover, plots of error asked as a function of error obtained show that for the orbits the curve is smoothly linear and for the decreasing exponential the curve is a wild step-like function.

```

PROCEDURE COWELL(N,XI,XIL,XF,XFL,Y,YL,F,EA,ER,P,DX,RKSFN,RKSORDER 00684000
,RKSCOEFF,RKSCOEFFL,Q,COWELLCOEFF,COWELLCOEFFL,START,DSHANKS); 00685000
VALUE N,XI,XIL,XF,XFL,P,DX,RKSFN,RKSORDER,Q; 00686000
INTEGER N,RKSFN,RKSORDER,Q; 00687000
REAL XI,XIL,XF,XFL,P,DX; 00688000
REAL ARRAY Y,YL,EA,ER,RKSCOEFF,RKSCOEFFL,COWELLCOEFF,COWELLCOEFFL 00689000
[0]; 00690000
PROCEDURE F,DSHANKS; 00691000
INTEGER PROCEDURE START; 00692000

BEGIN 00693000
  INTEGER C1,M,MM1,QM1,QP1,TQP1,INDX,I1,I2,I3,I4,I,J,K,CYI; 00694000
  INTEGER CORRECTCNT; 00695000
  REAL INT,INTL,C2,C2L,DFACTOR,X,XL,H,HL,T1,T1L,T2,T2L,T3,T3L,T4, 00696000
  T4L,T5,T5L,T6,T6L; 00697000
  BOOLEAN DFLAG,PFLAG; 00698000
  REAL ARRAY FH,FHLC,Q +Q,0:NJ,YMID1,YMID1L,YP,YPL,YC,YCL,YM,YML 00699000
  ,CS,CSL,FP,FPL,HDM1F,HDM1FL,HDM1FMID,HDM1FMIDL,EAV,ERV,EAVD,ERVD 00700000
  ,I0:NJ,PCOEFF,PCOEFFL,CCOEFF,CCOEFFL,MCOEFF,MCOEFFL,QJ; 00701000
  LABEL L0,L1,L2,L3,L4,L5,STARTER,DOUBLER,ACCEPT,CORRECT,CLOSER; 00702000
  DOUBLE(XF,XFL,XI,XIL,=,INT,INTL); 00703000
  C1:=1; 00704000
  L0:C1:=C1+C1; 00705000
  IF C1<Q THEN GO TO L0; 00706000
  IF DX<0 THEN DX:=-DX; 00707000
  IF DX#0 THEN 00708000
  BEGIN 00709000
    L1:IF ABS(INT)/C1>DX THEN 00710000
    BEGIN 00711000
      C1:=C1+C1; 00712000
      GO TO L1 00713000
    END 00714000
  END 00715000
END; 00716000
DOUBLE(C1,0,1.0,X,=,C2,C2L); 00717000
M:=Q DIV 2; 00718000
MM1:=M-1; 00719000
QM1:=Q-1; 00720000
QP1:=Q+1; 00721000
TQP1:=QP1+Q; 00722000
DFACTOR:=2.0*(Q+3); 00723000

```



```

T2L := MC0EFFFL[I4];
FOR J := 1 STEP 1 UNTIL N DO DOUBLE(HDM1F[J], HDM1FL[J], FH[I1, J], FH[I1, J],
    FHL[I1, J], T1L, X, FHL[I2, J], FHL[I2, J], T2L, X, HDM1F[J], HDM1FL[J],
    J, HDM1FL[J])
END;
FOR J := 1 STEP 1 UNTIL N DO
BEGIN
HDM1FMIDL[J] := HDM1F[J];
HDM1FMIDL[J] := HDM1FL[J];
END;
DFLAG := FALSE;
ACCEPT: FOR I := 1 STEP 1 UNTIL M DO
BEGIN
CYI := (INDEX := (INDEX + 1) MOD TQP1) + TQP1;
DOUBLE(XF, XFL, C2, C2L, I, 0, H, H, X, X, X, XL);
I1 := (CYI - 1) MOD TQP1;
T1 := PC0EFFFL[I];
T1L := PC0EFFFL[0];
T2 := CC0EFFFL[1];
T2L := CC0EFFFL[1];
FOR J := 1 STEP 1 UNTIL N DO
BEGIN
T3 := FH[I1, J];
T3L := FHL[I1, J];
DOUBLE(H, HL, T3, T3L, X, HDM1F[J], HDM1FL[J], T4, T4L);
HDM1F[J] := T4;
HDM1FL[J] := T4L;
DOUBLE(I1, T1L, T3, T3L, X, T4, T4L, YP[J], YPL[J]);
DOUBLE(T2, T2L, T3, T3L, X, T4, T4L, CS[J], CSL[J]);
END;
I3 := CYI - QP1;
FOR K := 2 STEP 1 UNTIL M DO
BEGIN
I1 := (CYI - K) MOD TQP1;
I2 := (I3 + K) MOD TQP1;
T1 := PC0EFFFL[I4 := K - 1];
T1L := PC0EFFFL[I4];
T2 := CC0EFFFL[K];
T2L := CC0EFFFL[K];
T3 := PC0EFFFL[I4 := Q - K];

```

```

T3L :=PCOEFFL[I4];
T4 :=CCOEFFL[I4 :=QP1 -K];
T4L :=CCOEFFL[I4];
FOR J :=1 STEP 1 UNTIL N DO
BEGIN
  T5 :=FHL[I1,J];
  T5L :=FHL[I1,J];
  T6 :=FHL[I2,J];
  T6L :=FHL[I2,J];
  DOUBLE(T1,T1L,T5,T5L,x,T3,T3L,T6,T6L,x,+,YP[J],YPL[J],+,:=0813000
  ,YP[J],YPL[J]);
  DOUBLE(T2,T2L,T5,T5L,x,T4,T4L,T6,T6L,x,+,CS[J],CSL[J],+,:=0815000
  ,CSL[J],CSL[J])
END
END ;
I1 :=(CYI -Q)MOD TQP1 ;
I2 :=(CYI -QP1)MOD TQP1 ;
T1 :=PCUEFFL[Q1];
T1L :=PCOEFFL[Q1];
T2 :=CCUEFFL[Q];
T2L :=CCOEFFL[Q];
T3 :=PCUEFFL[Q];
T3L :=PCOEFFL[Q];
FOR J :=1 STEP 1 UNTIL N DO
BEGIN
  T4 :=FHL[I1,J];
  T4L :=FHL[I1,J];
  DOUBLE(T1,T1L,T4,T4L,x,T3,T3L,FHL[I2,J],FHL[I2,J],x,+,YP[J],
  YPL[J],+,:=,YP[J],YPL[J]);
  DOUBLE(T2,T2L,T4,T4L,x,CS[J],CSL[J],+,:=,CS[J],CSL[J])
END ;
T2 :=CCOEFFL[0];
T2L :=CCOEFFL[0];
CORRECT:=1 ;
CORRECT:=F(N,X,XL,YP,YPL,FP,FPL);
FOR J :=1 STEP 1 UNTIL N DO
BEGIN
  DOUBLE(T2,T2L,FP[J],FPL[J],x,CS[J],CSL[J],+,:=,T3,T3L);
  YC[J]:=T3 ;
  YCL[J]:=T3L ;

```

```

DOUBLE(T3,T3L,YP[J],YPL[J],, ,:=,T1,T1L);
IF (T1 :=ABS(T1))>EAV[J]THEN
BEGIN
  IF T1 >ERV[J]*ABS(T3)THEN
  BEGIN
    FOR J :=J ÷1 STEP 1 UNTIL N DO DOUBLE(T2,T2L,FP[J],FPL[J],X,CSL[J],+ ,:=,YC[J],YCL[J]);
    J,X,CSL[J],YC[J],+ ,:=,YC[J],YCL[J];
    FCN,X,XL,YC,YCL,FP,FPL;
    FOR J :=1 STEP 1 UNTIL N DO
    BEGIN
      DOUBLE(T2,T2L,FP[J],FPL[J],X,CSL[J],+ ,:=,T3,T3L);
      YP[J]:=T3 ;
      YPL[J]:=T3L ;
      DOUBLE(T3,T3L,YC[J],YCL[J],+ ,:=,T1,T1L);
      IF T1 >ERV[J]*ABS(T3)THEN
      BEGIN
        IF (T1 :=ABS(T1))>EAV[J]THEN
        BEGIN
          FOR J :=J ÷1 STEP 1 UNTIL N DO DOUBLE(T2,T2L,FP[J],FPL[J],X,CSL[J],+ ,:=,YP[J],YPL[J]);
          CORRECTCNT :=CORRECTCNT +2 ;
          IF CORRECTCNT >8 THEN
          BEGIN
            INDX :=(CYI -I)MOD TQP1 ;
            GO TO L5
          END ;
          GO TO CORRECT
        END
      END
    END ;
    FCN,X,XL,YC,YPL,FH[INDX,*],FHL[INDX,*]);
    PFLAG :=TRUE ;
    CORRECTCNT :=CORRECTCNT +1 ;
    GO TO L2
  END
END
END ;
FCN,X,XL,YC,YCL,FH[INDX,*],FHL[INDX,*]);
PFLAG :=FALSE ;

```

```

L2:
END ;
I1 :=(CYI -M)MOD TQP1 ;
T1 :=MCOEFFFL[M];
T1L :=MCOEFFFL[M];
FOR J :=1 STEP 1 UNTIL N DO DOUBLE(T1,T1L,FH[I1,J],FHL[I1,J],X,
HDM1FMID[J],HDM1FMIDL[J],+,:=,YM[J],YML[J]);
I3 :=CYI -Q ;
FOR K :=0 STEP 1 UNTIL MM1 DO
BEGIN
I1 :=(CYI -K)MOD TQP1 ;
I2 :=(I3 +K)MOD TQP1 ;
T1 :=MCOEFFFL[K];
T1L :=MCOEFFFL[K];
T2 :=MCOEFFFL[I4 :=Q -K];
T2L :=MCOEFFFL[I4];
FOR J :=1 STEP 1 UNTIL N DO DOUBLE(T1,T1L,FH[I1,J],FHL[I1,J],X,
,T2,T2L,FH[I2,J],FHL[I2,J],+,:=,YM[J],YML[J]
)
END ;
IF DFLAG THEN
BEGIN
FOR J :=1 STEP 1 UNTIL N DO
BEGIN
T3 :=Y[J];
DOUBLE(T3,YL[J],YM[J],YML[J],-,:=,T2,T2L);
IF (T2 :=ABS(T2))>EAVD[J]THEN
BEGIN
IF T2 >ERVDI[J]*ABS(T3)THEN
BEGIN
IF T2 >EAV[J]THEN
BEGIN
IF T2 >ERV[J]*ABS(T3)THEN GO TO L4
END ;
GO TO L3
END
END
END ;
DOUBLE(C2,C2L,M,0,+,:=,C2,C2L);
DOUBLE(C2,C2L,2,0,/,:=,C2,C2L);

```

```

00884000
00885000
00886000
00887000
00888000
00889000
00890000
00891000
00892000
00893000
00894000
00895000
00896000
00897000
00898000
00899000
00900000
00901000
00902000
00903000
00904000
00905000
00906000
00907000
00908000
00909000
00910000
00911000
00912000
00913000
00914000
00915000
00916000
00917000
00918000
00919000
00920000
00921000
00922000
00923000

```

```

IF PFLAG THEN FOR J :=1 STEP 1 UNTIL N DO
BEGIN
  YL[J]:=YP[J];
  YL[J]:=YPL[J]
END ELSE FOR J :=1 STEP 1 UNTIL N DO
BEGIN
  YL[J]:=YC[J];
  YL[J]:=YCL[J]
END ;
C1 :=C1 DIV 2 ;
IF C2 <M THEN GO TO CLOSER ;
INDX :=INDX +1 ;
FOR K :=1 STEP 1 UNTIL 0 DO
BEGIN
  INDX :=(INDX +1)MOD TQP1 ;
  I1 :=(INDX +K)MOD TQP1 ;
  FOR J :=1 STEP 1 UNTIL N DO
  BEGIN
    FHL[INDX,J]:=FH[I1,J];
    FHL[INDX,J]:=FHL[I1,J]
  END
END ;
GO TO DOUBLER
END ;
J:=0 ;
L3:FOR J :=J +1 STEP 1 UNTIL N DO
BEGIN
  T3 :=Y[J];
  DOUBLE(T3,YL[J],YM[J],YML[J],:=,T2,T2L);
  IF (T2 :=ABS(T2))>EAV[J]THEN
  BEGIN
    IF T2 >ERV[J]*ABS(T3)THEN
    BEGIN
      L4:INDX :=(CYI -M)MOD TQP1 ;
      L5:C1 :=C1 +C1 ;
      DOUBLE(XF,XFL,C2,C2L,H,HL,X,:=,X,XL);
      DOUBLE(C2,C2L,C2,C2L,+,:=,C2,C2L);
      GO TO STARTER
    END
  END
END

```


D. The Runge-Kutta-Shanks Method

1. Introduction

The procedure described is a generalization of the Runge-Kutta method for solving a system of differential equations. It may be applied to an arbitrary system of first-order differential equations of the form

$$\vec{y}' = \vec{f}(x, \vec{y})$$

with the initial conditions

$$\vec{y}(x_0) = \vec{y}_0$$

$$\text{where } \vec{y}(x) = \begin{pmatrix} y_1(x) \\ \vdots \\ y_n(x) \end{pmatrix}, \quad \vec{y}'(x) = \begin{pmatrix} y_1'(x) \\ \vdots \\ y_n'(x) \end{pmatrix},$$

$$\vec{f}(x, \vec{y}) = \begin{pmatrix} f_1(x, y_1, \dots, y_n) \\ \vdots \\ f_n(x, y_1, \dots, y_n) \end{pmatrix}, \quad \vec{y}_0 = \begin{pmatrix} y_{10} \\ \vdots \\ y_{n0} \end{pmatrix}.$$

2. Description of the Method

The Shanks Method is a single-step procedure for finding a numerical solution of a first-order ordinary differential equation or system of differential equations in which the derivatives of the dependent variables may be expressed explicitly as functions of the independent and dependent variables.

Consider the system of differential equation

$$\vec{y}' = \vec{f}(x, \vec{y}).$$

Suppose the value of $\vec{y}(x)$ is known. The value $\vec{y}(x+h)$ is approximated by

$$\vec{y}(x+h) = \vec{y}(x) + h \sum_{i=1}^m \gamma_i \vec{f}_i(x, h, \vec{y}),$$

where

$$\vec{f}_1(x, h, \vec{y}) = \vec{f}(x, \vec{y}),$$

$$\vec{f}_i(x, h, \vec{y}) = \vec{f}\left(x + \alpha_i h, \vec{y} + h \sum_{j=1}^{i-1} \beta_{ij} \vec{f}_j\right), i = 2, \dots, m.$$

The coefficients α_i ($i = 2, \dots, m$),

$$\beta_{ij} \text{ (} i = 2, \dots, m; j = 1, \dots, i-1 \text{), and } \gamma_i \text{ (} i = 1, \dots, m \text{)}$$

are chosen so as to make the approximation correct to some order. A

special case of the Shanks formula is the fourth-order Runge-Kutta formula:

$$\alpha_2 = 1/2, \alpha_3 = 1/2, \alpha_4 = 1,$$

$$\beta_{21} = 1/2, \beta_{31} = 0, \beta_{32} = 1/2, \beta_{41} = \beta_{42} = 0, \beta_{43} = 1,$$

$$\gamma_1 = 1/6, \gamma_2 = 1/3, \gamma_3 = 1/3, \gamma_4 = 1/6.$$

For useful values of the various combinations of α , β , and γ , see Shanks [].

3. The Computer Procedure

The procedure was programmed for the B-5500 computer in the B-5500 Algol language. Double-precision arithmetic (22 decimal digits) was used. This is a double-precision version of the single-precision procedure described in the previous report [21].

3.1 Error Estimates and Step Size Control

In this procedure a single set of Shanks formulas is used.

Suppose a vector $\vec{y}(x)$ is known. Then the Shanks method is applied to one step of size h (where $h = \frac{\Delta x}{c}$, Δx is the length of the interval, and c is a power of two), and to two steps of size $\frac{h}{2}$, as follows:

$$\vec{y}_p = \vec{y}(x) + h \sum_{i=1}^m \gamma_i \vec{f}_i(x, h, \vec{y}),$$

$$\vec{y}_m = \vec{y}(x) + \frac{h}{2} \sum_{i=1}^m \gamma_i \vec{f}_i(x, \frac{h}{2}, \vec{y})$$

$$\vec{y}_c = \vec{y}_m + \frac{h}{2} \sum_{i=1}^m \gamma_i \vec{f}_i(x + \frac{h}{2}, \frac{h}{2}, \vec{y}_m).$$

Both \vec{y}_p and \vec{y}_c are estimates of $\vec{y}(x+h)$. An error estimate $E_k = |y_{ck} - y_{pk}|$ is calculated for each independent variable y_k . If both $E_k > \frac{E_{ak}}{c^p}$ and $E_k > \frac{E_{rk}|y_{ck}|}{c^p}$ for any dependent variable where E_{ak} is an absolute error estimate, E_{rk} is a relative error estimate, and p is an input parameter, then the step is rejected and the step size is halved; otherwise the step is accepted and \vec{y}_c is taken as the vector $\vec{y}(x+h)$. If for every dependent variable, either $E_k < \frac{E_{ak}}{2^{(j+3)}c^p}$ or $E_k < \frac{E_{rk}|y_{ck}|}{2^{(j+3)}c^p}$, where j is the order, then the step size is doubled. If the step size h is larger than the distance to the end of the interval, then that distance is taken as the step size.

3.2 Input and Output of the Procedure

The procedure is called as follows:

DSHANKS (N, XI, XII, XF, XFL, YV, YVL, F, M, ORDER, CF, CFL, F, EA, ER, DX);

where the parameters have the following meaning:

N - number of dependent variables;

XI - high part of the initial value of the independent variable;

XIL - low part of the initial value of the independent variable;

XF - high part of the final value of the independent variable;

XFL - low part of the final value of the independent variable;

YV - array of the high parts of the initial values of the dependent variables, based at zero but with the zero element not used;

YVL - array of the low parts of the initial values of the dependent variables;

F - a function evaluation procedure, supplied by the user, called as follows:

F (N, X, XL, YV, YVL, FV, FVL);

where N is the number of dependent variables, X is the high part of the value of the independent variable, XL is the low part of the value of the independent variable, YV is the array of the high parts of the values of the dependent variables, YVL is the array of the low parts of the values of the dependent variables, FV is the array in which the high parts of the function values are placed, and FVL is the array in which the low parts of the function values are placed;

M - the number of function evaluations in each application of the Runge-Kutta-Shanks method;

ORDER - the order of the Shanks formulas used;

CF - the array of the high parts of the Runge-Kutta-Shanks coefficients, starting in the zero element arranged as follows: for each i, the corresponding $\alpha_i \beta_{ij}$'s, followed by α_i with the γ_i 's at the end;

CFL - the array of the low parts of the Runge-Kutta-Shanks coefficients;

P - an exponent used in step size control;

EA - an array of absolute errors asked;

ER - an array of relative errors asked;

DX - a recommended starting step size (the actual starting step size will be the largest binary fraction of the total interval which is not larger than DX);

The high and low parts of the final values of the dependent variables are stored in YV and YVL, respectively, before exiting the procedure.

4. Flow Diagram and Program Listing

Figure 4 is the flow diagram for the Runge-Kutta-Shanks procedure.

A listing of the program is given at the end of this section.

5. Results and Conclusions

The Runge-Kutta-Shanks double-precision procedure was first used with the three-body problem and the input parameter $p = 1/2$, controlling on absolute and relative error simultaneously.

At first, the error estimate E_k described in paragraph 3.1 above was computed as

$$E_k = \frac{|y_{ck} - y_{pk}|}{2^{\text{order} - 1}}$$

as had been done in single-precision experiments [21]. However, it was found that for some orders the accuracy asked was not achieved. It was then decided to change the error estimate to

$$E_k = \left| y_{ck} - y_{pk} \right| .$$

In this case, the accuracy asked was consistently achieved, and for some orders (6-6, 7-9, 8-10, 8-12) the accuracy obtained was consistently better than that asked. It was found that for the three-body problem the most accurate formulas were the 6-6 and 8-10, with the 7-9 and 8-12 slightly less accurate, and the 4-4, 5-5, and 7-7 achieving only the accuracy asked. The 8-12 formula was faster, the 7-9 and 8-10 almost as fast, the 5-5, 6-6, and 7-7 considerably slower, and the 4-4 extremely slow. This indicates that the classical Runge-Kutta 4-4 formula is not suitable for double precision. It is recommended that only the 7-9, 8-10, and 8-12 formulas be used in double-precision for the three-body problem.

The procedure was then used with the differential equation $y' = -y$ (i.e., the equation for the negative exponential), using the input parameters $p = 0$, $p = 1/2$, and $p = 1$, controlling on absolute and relative error separately.

Controlling on absolute error, the procedure always achieved at least an order of magnitude more accuracy than was asked, even with $p = 0$. Controlling on relative error, the 4-4, 5-5, and 7-7 formulas sometimes failed to achieve the accuracy asked with $p = 0$ and occasionally with $p = 1/2$. In most cases the 7-7 was fastest for low accuracies and the 8-10 for high accuracies. Again the 4-4 was slowest. For the higher accuracies the 5-5 was also very slow. It is recommended that for double precision the 4-4 and 5-5 formulas be excluded from consideration. The

6-6 might also be excluded, since it rarely was faster than the higher-order formulas.

For high accuracies, it appears that, in general, the higher the order, the more suitable a set of formulas is, although this is not an absolute rule. Between formulas of the same order, sometimes one might be more suitable and sometimes the other. For instance, the 8-10 was faster for the negative exponential case and the 8-12 for the three-body problem.

Further work remains to be done in this area. A wider range of problems might be considered for experimentation with the procedure. If a higher-order set of formulas is available, it might be used. A procedure which selected an order, or which selects single or double precision according to the accuracy asked, might be developed. Further experimentation in step-size control, including the possibility of continuous step-size control, might be considered.


```

REAL C,CL;
REAL ARRAY FV,FVL,YV,YVL[0];
BEGIN
  INTEGER K;
  DEFINE CD=C,CL, FVK=FV[K],FVL[FVL[K]],YVK=YV[K],YVL[YVL[K]];
  FOR K←0 STEP 1 UNTIL N DO DOUBLE(FVK,CD, X, YVK, ←, YVK);
END;
M←M-1;
DOUBLE(XFD,XID, ←, DXTD);
IF DXI=0 THEN GO TO EXIT;
DXD←DXI;
DXDL←DXTL;
C1←1;
IF DX≠0 THEN
  BEGIN
    WHILE ABS(DX)<ABS(DXD) DO
      BEGIN
        C1←C1+C1;
        DOUBLE(DXTD,C1D, ←, DXDD);
      END;
    END;
    TWOC1←C1+C1;
    C2←C1;
    C2L←0;
    DOUBLE(DXTD,TWOC1D, ←, DXHD);
    ERANGE←1/2*(ORDER+3);
    EFACOR←C1+P;
    NCF1←(MXM+M)DIV 2+M+M;
    NCF←NCF1+1;
    CFSW←FALSE;
    FOR I←0 STEP 1 UNTIL NCF1 DO
      BEGIN
        J←I+NCF;
        DOUBLE(CFI,DXDD, X, ←, CI);
        DOUBLE(CFI,DXHD, X, ←, CJ);
      END;
    END;
  END;

```

01041000
01042000
01043000
01044000
01045000
01046000
01047000
01048000
01049000
01050000
01051000
01052000
01053000
01054000
01055000
01056000
01057000
01058000
01059000
01060000
01061000
01062000
01063000
01064000
01065000
01066000
01067000
01068000
01069000
01070000
01071000
01072000
01073000
01074000
01075000
01076000
01077000
01078000
01079000
01080000

```

END;
DOUBLE(XID,D1DXHD,D2←XMD);
L1:DSW←TRUE;
F(N,XD,YVD,GVD);
IF CFSW THEN L←NCF1 ELSE L←-1;
FOR I←1STEP 1UNTIL M DO
BEGIN
K←I-1;
L←L+1;
ADD1(N,CDL,GVD,YVD,YPD);
FOR J←1STEP 1UNTIL K DO
BEGIN
L←L+1;
ADD2(N,CDL,FVJ,YPD);
END;
L←L+1;
DOUBLE(CDL,XD,D1←TEMPD);
F(N,TEMPD,YPD,FVI);
END;
L←L+1;
ADD1(N,CDL,GVD,YVD,YPD);
FOR I←1STEP 1UNTIL M DO
BEGIN
L←L+1;
ADD2(N,CDL,FVI,YPD);
END;
L2:IF CFSW THEN L←-1;
FOR I←1STEP 1UNTIL M DO
BEGIN
K←I-1;
L←L+1;
ADD1(N,CDL,GVD,YVD,YMD);
FOR J←1STEP 1UNTIL K DO
BEGIN
L←L+1;
ADD2(N,CDL,FVJ,YMD);

```

```

01081000
01082000
01083000
01084000
01085000
01086000
01087000
01088000
01089000
01090000
01091000
01092000
01093000
01094000
01095000
01096000
01097000
01098000
01099000
01100000
01101000
01102000
01103000
01104000
01105000
01106000
01107000
01108000
01109000
01110000
01111000
01112000
01113000
01114000
01115000
01116000
01117000
01118000
01119000
01120000

```

```

END;
L←L+1;
DOUBLE(CDL,XD,TEMPD);
F(N,TEMPD,YMD,FVI);
END;
L←L+1;
ADD1(N,CDL,GVD,YVD,YMD);
FOR I←1STEP 1UNTIL M DO
BEGIN
L←L+1;
ADD2(N,CDL,FVI,YMD);
END;
F(N,XMD,YMD,FV0);
IF CFSW THEN L←1ELSE L←NCF;
FOR I←1STEP 1UNTIL M DO
BEGIN
K←I-1;
L←L+1;
ADD1(N,CDL,FV0,YMD,YCD);
FOR J←1STEP 1UNTIL K DO
BEGIN
L←L+1;
ADD2(N,CDL,FVJ,YCD);
END;
L←L+1;
DOUBLE(CDL,XMD,TEMPD);
F(N,TEMPD,YCD,FVI);
END;
L←L+1;
ADD1(N,CDL,FV0,YMD,YCD);
FOR I←1STEP 1UNTIL M DO
BEGIN
L←L+1;
ADD2(N,CDL,FVI,YCD);
END;

```

```

01121000
01122000
01123000
01124000
01125000
01126000
01127000
01128000
01129000
01130000
01131000
01132000
01133000
01134000
01135000
01136000
01137000
01138000
01139000
01140000
01141000
01142000
01143000
01144000
01145000
01146000
01147000
01148000
01149000
01150000
01151000
01152000
01153000
01154000
01155000
01156000
01157000
01158000
01159000
01160000

```

```

FOR K←1STEP 1UNTIL N DO
  BEGIN
    DOUBLE(CYK,YPK,←←TEMPD);
    IF TEMP#0THEN
      BEGIN
        ES←ABS(TEMP)×EFACOR;
        IF ES≥EAK1THEN IF ES≥ABS(YC1K1)×ER[K]THEN
          BEGIN
            DSW←FALSE;
            C1←TWOC1;
            TWOC1←C1+C1;
            DOUBLE(C2D,TWO×←←C2D);
            DXD←DXH;
            DXDL←DXHL;
            DOUBLE(DXTD,TWOC1D,←←DXHD);
            EFACOR←C1×P;
            IF CFSW THEN
              BEGIN
                CFSW←FALSE;
                FOR I←0STEP 1UNTIL NCF1 DO
                  BEGIN
                    J←I+NCF;
                    DOUBLE(CFI,DXHD,←←CJ);
                  END;
                END;
              END ELSE
                BEGIN
                  CFSW←TRUE;
                  FOR I←0STEP 1UNTIL NCF1 DO DOUBLE(CFI,DXHD,←←CI);
                END;
            DOUBLE(XFD,C2D,HALF,←←DXDD,←←←XMD);
            FOR K←1STEP 1UNTIL N DO
              BEGIN
                YP[K]←YM[K];
                YPL[K]←YML[K];
              END;
            END;
            GO TO L2;
          END;
        GO TO L2;
      END;
    END;
  END;

```

```

01201000
01202000
01203000
01204000
01205000
01206000
01207000
01208000
01209000
01210000
01211000
01212000
01213000
01214000
01215000
01216000
01217000
01218000
01219000
01220000
01221000
01222000
01223000
01224000
01225000
01226000
01227000
01228000
01229000
01230000
01231000
01232000
01233000
01234000
01235000
01236000
01237000
01238000
01239000
01240000

END;
IF DSW THEN IF ES≥EA[K]*ERANGE THEN IF ES≥ABS(YC[K])*ER[K]
XERANGE THEN DSW←FALSE;

END;

DOUBLE(C2D,ONE,*,C2D);
FOR K←1STEP 1UNTIL N DO
BEGIN
YV[K]←YC[K];
YVL[K]←YCL[K];

END;
IF C2=0THEN GO TO EXIT;
DOUBLE(XFD,C2D,DxDD,x,*,*,XD);
IF DSW THEN IF C2>1THEN
BEGIN
TWO C1←C1;
C1←C1 DIV 2;
DOUBLE(C2D,TWO,/,*,C2D);
DXH←DXD;
DXHL←DXDL;
DOUBLE(DXTD,C1D,/,*,DXDD);
EFACTOR←C1*P;
IF CFSW THEN
BEGIN
CFSW←FALSE;
FOR I←0STEP 1UNTIL NCF1 DO DOUBLE(CFI,DxDD,x,*,CI);

END ELSE
BEGIN
CFSW←TRUE;
FOR I←0STEP 1UNTIL NCF1 DO
BEGIN
J←I+NCF;
DOUBLE(CFI,DxDD,x,*,CJ);

END;

```

01241000
01242000
01243000
01244000
01245000
01246000
01247000
01248000
01249000
01250000
01251000
01252000
01253000
01254000
01255000
01256000
01257000
01258000
01259000
01260000
01261000
01262000
01263000
01264000
01265000
01266000
01267000
01268000
01269000
01270000
01271000
01272000
01273000

```

END;
IF C2<1 THEN
BEGIN
  DOUBLE(C2D,ONE,TEMPD);
  IF TEMP<0 THEN
  BEGIN
    EFACOR←(C1/C2)*P;
    C1←C2←1;
    C2L←0;
    TWOC1←2;
    DOUBLE(XFD,XD,DXTD);
    DXD←DXT;
    DXDL←DXTL;
    DOUBLE(DXDD,TWO,DXHD);
    CFSW←FALSE;
    FOR I←0 STEP 1 UNTIL NCF1 DO
    BEGIN
      J←I+NCF;
      DOUBLE(CFI,DXDD,X,CI);
      DOUBLE(CFI,DXHD,X,CJ);
    END;
  END;
END;
DOUBLE(XFD,C2D,HALF,DXDD,X,DXMD);
GO TO L1;
EXIT;
END;

```

E. The General Multistep Method Starting Procedure

1. Introduction

The general multistep method starting procedure is a B-5500 ALGOL double-precision Runge-Kutta-Shanks procedure used for obtaining starting values for the Adams, Butcher, and Cowell multistep methods. The declaration is as follows:

```
integer procedure start (n, xi, xil, xf, xfl, cl, ea, er, f, m, x,  
                        xl, yiv, yivl, yh, yhl, fh, fhl, yfv,  
                        yfv1, cyi, cym, pa, p, fneval, rksconst,  
                        rksconst1);
```

```
value n, xi, xil, xf, xfl, cl, m, cyi, cym, pa, p, fneval;
```

```
integer n, cl, m, cyi, cym, pa, fneval;
```

```
real xi, xil, xf, xfl, x, xl, p;
```

```
real array ea, er, yiv, yivl, yfv, yfv1, rksconst, rksconst1 [0],  
            yh, yhl, fh, fhl [0,0];
```

```
procedure f;
```

2. Description of the Procedure

The parameters of the procedure are defined as follows:

n - the number of dependent variables

xi, xil - the high and low halves, respectively, of the starting value of the independent variable x passed to the multistep method

xf, xfl - the high and low halves, respectively, of the final value of the independent variable x passed to the multistep method

cl - the integer counter (xf - xi)/h from the multistep method

ea - the absolute error vector passed to the multistep method
er - the relative error vector passed to the multistep method
f - the procedure which computes $\vec{f}(x, \vec{y}) = \vec{y}$
m - the number of history points to be calculated by start
x, xl - the high and low halves, respectively, of the value of the independent variable at which start begins its integration
yiv, yivl - the arrays which contain on entry for Adams and Cowell the high and low halves, respectively, of the values of the dependent variables at x and which contains on exit for Cowell the high and low halves, respectively, of the values of the dependent variables at the mth point calculated by start
yh, yh1 - the arrays which contain on entry for Butcher in row cyi the high and low halves, respectively, of the values of the dependent variables at x and which contain on exit for Butcher the high and low halves, respectively, of the values of the dependent variables at each of the m points calculated by start
fh, fh1 - the arrays which contain on entry in row cyi the high and low halves, respectively, of the function values at x and which contain on exit the high and low halves, respectively, of the function values at each of the m points calculated by start
yfv, yfv1 - the arrays which contain on exit the high and low halves, respectively, of the values of the dependent variables at the mth point calculated by start for Adams or the m/2th point calculated by start for Cowell

cyi - the cyclic index indentifying on entry the row of yh and yh1 in which the values of the dependent variables at x are stored for Butcher and the row of fh and fh1 in which the function values at x are stored for any method

cym - the number of rows in the arrays yh, yh1, fh, and fh1

pa - the parameter which is zero for Adams, one for Cowell, two for Butcher

p - the exponent such that the absolute error at each step is not to exceed $\underline{ea}/\underline{cl}^p$ and the relative error at each step is not to exceed $\underline{er}/\underline{cl}^p$

fneval - the number of function evaluations required by the Runge-Kutta-Shanks procedure

rksconst, rksconst1 - the arrays which contain the high and low halves, respectively, of the Runge-Kutta-Shanks coefficients in the same order as required by the procedure dshanks described in section D.

The value of start on exit is two to the power of the number of halvings which took place within start.

Although the base of the arrays ea, er, yiv, yiv1, yfv, and yfv1 and of the rows of yh, yh1, fh, and fh1 is zero, the n components are placed in positions 1, 2, . . . , n and the zero position is unused.

The procedure attempts to calculate m (if m is even and positive) or m + 1 (if m is odd) Runge-Kutta-Shanks steps of size $h = (\underline{xf} - \underline{xi})/\underline{cl}$. After each even step of size h is taken, one step of size $2h$ is taken over the interval spanned by the two steps of size h . The absolute value of the differences in each dependent variable between the $2h$ -step and the second h -step is compared with the corresponding component of $\vec{ea}/(\underline{cl}/2)^p$

for absolute error and with the product of the corresponding component of $\vec{er}/(\underline{cl}/2)^{\underline{p}}$ and the corresponding dependent variable value from the second h-step for relative error. If each component of the difference does not exceed in either the absolute or the relative error test and \underline{m} steps have not yet been taken, the process of two h-steps, one 2h-step, and test is continued. If any component of the difference exceeds in both the absolute and the relative error tests, \underline{cl} is doubled, h is halved, and integration begins at \underline{x} . The first step of previous size h was saved and becomes the first step of present size $2h$.

The \underline{m} calculated function values from h-steps are placed in rows $(\underline{cyi} + 1) \bmod \underline{cym}$, $(\underline{cyi} + 2) \bmod \underline{cym}$, . . . , $(\underline{cyi} + \underline{m}) \bmod \underline{cym}$ of the array \underline{fh} . For Butcher, the corresponding dependent variable values from h-steps are placed in the corresponding rows of the array \underline{yh} ; if \underline{m} is odd, the values of the dependent variable after h-step $\underline{m} + 1$ are placed in row $(\underline{cyi} + \underline{m} + 1) \bmod \underline{cym}$ of \underline{yh} . For Adams, the dependent variable values from h-step \underline{m} are placed in the array \underline{yfv} . For Cowell, the dependent variable values from h-step \underline{m} are placed in the array \underline{yiv} and from h-step $\underline{m}/2$ (\underline{m} is always even for Cowell) are placed in \underline{yfv} . If \underline{m} is zero, no calculation takes place.

3. Flow Diagram and Program Listing

Figure 5 is the flow diagram for the starting procedure. The program listing follows at the end of this section.

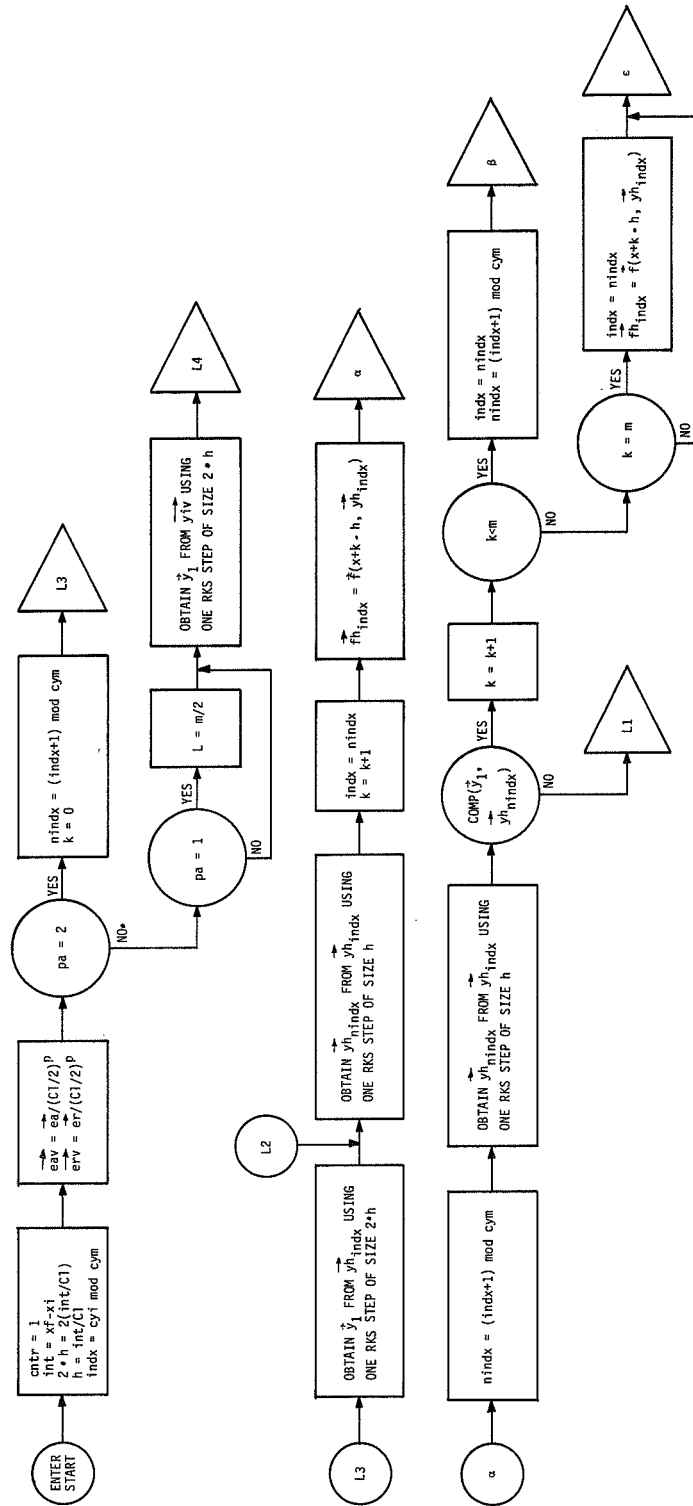


Figure 5. Flow Diagram for the General Multistep Method Starting Procedure.

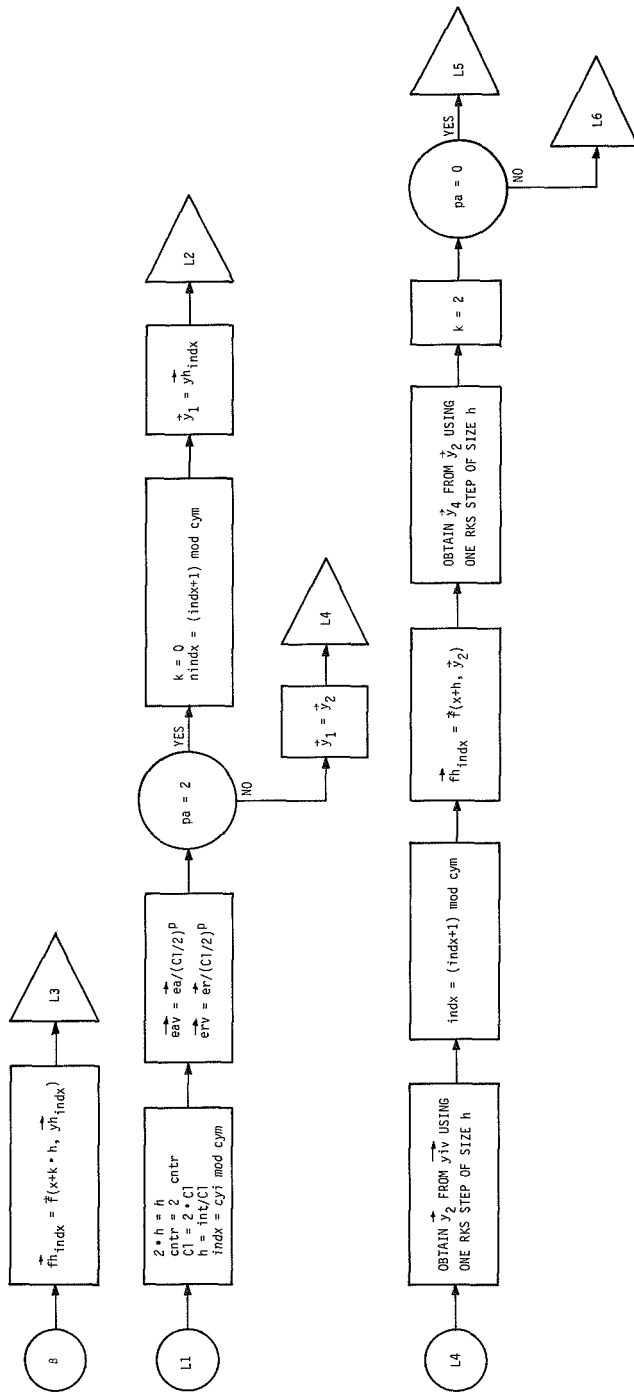
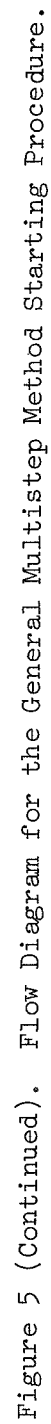


Figure 5 (Continued). Flow Diagram for the General Multistep Method Starting Procedure.



```

INTEGER PROCEDURE DSTART(N,XI,XIL,XF,XFL,C1,EA,ER,F,M,X,XL,YIV,
YIVL,YH,YHL,FH,FHL,YFV,YFVL,CYI,CYM,PA,P,FNEVAL,RKCONSTL,RKSCNSTL
),
VALUE N,XI,XIL,XF,XFL,C1,M,CYI,CYM,PA,P,FNEVAL,
INTEGER N,C1,M,CYI,CYM,PA,P,FNEVAL,
REAL XI,XIL,XF,XFL,X,XL,P,
REAL ARRAY EA,ER,YIV,YIVL,YFV,YFVL,RKCONSTL,RKSCNSTL(0),YH,YHL,FH00022000
,FHL(0,0),
PROCEDURE F,
BEGIN
  INTEGER I,J,K,L,COEFFCNT,FNMAX,INDX,NINDX,CNTR,
  REAL INT,INTL,H,HL,TWOH,TWOHL,T1,T1L,XT,XTL,
  REAL ARRAY HC,HCL,TWOHC,TWOHCL(0),((FNEVAL*3)*FNEVAL)/2,EAV
  ,ERV,Y1,Y1L,Y2,Y2L,Y3,Y3L,Y4,Y4L(0,N),G,GL(0,FNEVAL*2,0,N),
  LABEL L1,L2,L3,L4,L5,L6,
  PROCEDURE RUNKUT(N,X,XL,FNMAX,COEFF,COEFFL,YIV,YIVL,YFV,YFVL,FV
  ,FVL,F,G,GL),
  VALUE N,X,XL,FNMAX,
  INTEGER N,FNMAX,
  REAL X,XL,
  REAL ARRAY COEFF,COEFFL,YIV,YIVL,YFV,YFVL(0),G,GL(0,0),
  PROCEDURE F,
BEGIN
  INTEGER I,J,K,CNTR,
  REAL TEMP,TEMPL,
  TEMP:=COEFF[CNTR:=0],
  TEMPL:=COEFFL[CNTR],
  FOR I:=0STEP 1UNTIL FNMAX DO
    BEGIN
      FOR J:=1STEP 1UNTIL N DO DOUBLE(FV[J],FVL[J],TEMP,TEMPL,X,
YIV[J],YIVL[J],+,:=, YFV[J],YFVL[J]),
      FOR K:=0STEP 1UNTIL I-1DO
        BEGIN
          TEMP:=COEFF[CNTR:=CNTR+1],
          TEMPL:=COEFFL[CNTR],

```

```

FOR J:=1STEP 1UNTIL N DO DOUBLE(G[K,J],GL[K,J],TEMP,TEMPL,00053000
  x,YFV[J],YFVL[J],+,:=,YFV[J],YFVL[J])
END;
CCNT:=CCNT+1;
DOUBLE(X,XL,COEFF[CCNT],COEFFL[CCNT],+,:=,TEMP,TEMPL);
F(N,TEMP,TEMPL,YFV,YFVL,G[I,*],GL[I,*]);
TEMP:=COEFF[CCNT]=CCNT+1;
TEMPL:=COEFFL[CCNT]
END;
FOR J:=1STEP 1UNTIL N DO DOUBLE(FV[J],FVL[J],TEMP,TEMPL,x,YIV
  [J],YIVL[J],+,:=,YFV[J],YFVL[J]);
FOR K:=0STEP 1UNTIL FNMAX DO
  BEGIN
    TEMP:=COEFF[CCNT]=CCNT+1;
    TEMPL:=COEFFL[CCNT];
    FOR J:=1STEP 1UNTIL N DO DOUBLE(G[K,J],GL[K,J],TEMP,TEMPL,x
      ,YFV[J],YFVL[J],+,:=,YFV[J],YFVL[J])
    END;
  END;
END;
BOOLEAN PROCEDURE COMP(N,EAV,ERV,Y,YL,Z,ZL);
VALUE N;
INTEGER N;
REAL ARRAY EAV,ERV,Y,YL,Z,ZL[0];
BEGIN
  INTEGER J;
  REAL T1,T1L;
  LABEL L1;
  FOR J:=1STEP 1UNTIL N DO
    BEGIN
      DOUBLE(Y[J],YL[J],Z[J],ZL[J],-,:=,T1,T1L);
      IF(T1:=ABS(T1))>EAV[J]THEN
        BEGIN
          IF T1>ERV[J]*ABS(Z[J])THEN
            BEGIN
              COMP:=FALSE;
              GO TO L1
            END
          END
        END
      END
    END
  END
END

```

```

END;
COMP:=TRUE;
L1:
END;
CNTR:=1;
IF M#0 THEN
BEGIN
COEFFCNT:=((FNEVAL+3)*FNEVAL)/2)-2;
FNMAX:=FNEVAL-2;
DOUBLE(XF,XFL,XI,XIL,=:=INT,INTL);
DOUBLE(INT,INTL,INT,INTL,+=C1,Q,/=:=TWOH,TWOHL);
DOUBLE(INT,INTL,C1,Q,/=:=H,HL);
FOR I:=0 STEP 1 UNTIL COEFFCNT DO
BEGIN
T1:=RKSCNST[I];
T1L:=RKSCNSTL[I];
DOUBLE(T1,T1L,H,HL,X,:=HCL[I],HCL[I]);
DOUBLE(T1,T1L,TWOH,TWOHL,X,:=TWOHCL[I],TWOHCL[I]);
END;
INDX:=CY1 MOD CYM;
T1:=(C1/2)*P;
FOR J:=1 STEP 1 UNTIL N DO
BEGIN
EAV[J]:=EAV[J]/T1;
ERV[J]:=ERV[J]/T1
END;
IF PA=2 THEN
BEGIN
NINDX:=(INDX+1) MOD CYM;
K:=0;
XT:=X;
XTL:=XL;
GO TO L3
END;
IF PA=1 THEN L:=M DIV 2;
RUNKUT(N,X,XL,FNMAX,TWOHC,TWOHCL,YIV,YIVL,Y1,Y1L,FH[INDX,*]),
FHL[INDX,*],F,G,GL);
GO TO L4;
L1:TWOH:=H;
TWOHL:=HL;
00093000
00094000
00095000
00096000
00097000
00098000
00099000
00100000
00101000
00102000
00103000
00104000
00105000
00106000
00107000
00108000
00109000
00110000
00111000
00112000
00113000
00114000
00115000
00116000
00117000
00118000
00119000
00120000
00121000
00122000
00123000
00124000
00125000
00126000
00127000
00128000
00129000
00130000
00131000
00132000

```

```

CNTR:=CNTR+CNTR;
C1:=C1+C1;
DOUBLE(INT,INTL,C1,0,/,:=,H,HL);
FOR I:=OSTEP 1 UNTIL COEFFCNT DO
BEGIN
  TWOHC[I]:=HC[I];
  TWOHCL[I]:=HCL[I];
  DOUBLE(RKSCONST[I],RKSCONSTL[I],H,HL,*,:=,HC[I],HCL[I]);
END;
INDX:=CYI MOD CYM;
T1:=(C1/2)*P;
IF PA=2 THEN
BEGIN
  K:=0;
  NINDX:=(INDX+1)MOD CYM;
  FOR J:=1 STEP 1 UNTIL N DO
  BEGIN
    EAV[J]:=EA[J]/T1;
    ERV[J]:=ER[J]/T1;
    Y1[J]:=YH[NINDX,J];
    Y1L[J]:=YHL[NINDX,J]
  END;
  L2:=DOUBLE(K,0,H,HL,*,X,XL,+,:=,XT,XTL);
  RUNKUT(N,XT,XTL,FNMAX,HC,HCL,YH[INDX,*],YHL[INDX,*],F,G,GL);
  K:=K+1;
  INDX:=NINDX;
  NINDX:=(INDX+1)MOD CYM;
  DOUBLE(K,0,H,HL,*,X,XL,+,:=,XT,XTL);
  F(N,XT,XTL,YH[INDX,*],YHL[INDX,*],FHL[INDX,*]);
  RUNKUT(N,XT,XTL,FNMAX,HC,HCL,YH[INDX,*],YHL[INDX,*],F,G,GL);
  K:=K+1;
  IF COMPC(N,EAV,ERV,Y1,Y1L,YH[NINDX,*],YHL[NINDX,*]) THEN
  BEGIN
    K:=K+1;
    IF K<M THEN
    BEGIN
      INDX:=NINDX;
      NINDX:=(INDX+1)MOD CYM;
      DOUBLE(K,0,H,HL,*,X,XL,+,:=,XT,XTL);
    END;
  END;

```

```

00133000
00134000
00135000
00136000
00137000
00138000
00139000
00140000
00141000
00142000
00143000
00144000
00145000
00146000
00147000
00148000
00149000
00150000
00151000
00152000
00153000
00154000
00155000
00156000
00157000
00158000
00159000
00160000
00161000
00162000
00163000
00164000
00165000
00166000
00167000
00168000
00169000
00170000
00171000
00172000

```

```

FC(N,X,T,XTL,YH[INDX,*],YHL[INDX,*],FHL[INDX,*],FHL[INDX,*])00173000
)
L3:RUNKUT(N,X,T,XTL,FNMAX,TWOHC,TWOHCL,YH[INDX,*],YHL
[INDX,*],Y1,Y1L,FHL[INDX,*],FHL[INDX,*],F,G,GL);
GO TO L2
END;
IF K=M THEN
BEGIN
INDX:=NINDX;
DOUBLE(K,0,H,HL,X,XL,+,:=,XT,XTL);
FC(N,X,T,XTL,YH[INDX,*],YHL[INDX,*],FHL[INDX,*],FHL[INDX,*])00183000
)
END
END ELSE GO TO L1
END ELSE
BEGIN
FOR J:=1STEP 1 UNTIL N DO
BEGIN
EAV[J]:=EAV[J]/T1;
ERV[J]:=ERV[J]/T1;
Y1[J]:=Y2[J];
Y1L[J]:=Y2L[J]
END;
L4:RUNKUT(N,X,XL,FNMAX,HC,HCL,YIV,YIVL,Y2,Y2L,FH[INDX,*],FHL
[INDX,*],F,G,GL);
INDX:=(INDX+1)MOD CYM;
DOUBLE(X,XL,H,HL,+,:=,XT,XTL);
FC(N,X,T,XTL,Y2,Y2L,FHL[INDX,*],FHL[INDX,*]);
RUNKUT(N,X,T,XTL,FNMAX,HC,HCL,Y2,Y2L,Y4,Y4L,FHL[INDX,*],FHL
[INDX,*],F,G,GL);
K:=2;
IF PA=0 THEN
BEGIN
L5:IF COMP(N,EAV,ERV,Y1,Y1L,Y4,Y4L) THEN
BEGIN
IF K<M THEN
BEGIN
INDX:=(INDX+1)MOD CYM;
DOUBLE(K,0,H,HL,X,XL,+,:=,XT,XTL);
FC(N,X,T,XTL,Y4,Y4L,FHL[INDX,*],FHL[INDX,*]);
00174000
00175000
00176000
00177000
00178000
00179000
00180000
00181000
00182000
00183000
00184000
00185000
00186000
00187000
00188000
00189000
00190000
00191000
00192000
00193000
00194000
00195000
00196000
00197000
00198000
00199000
00200000
00201000
00202000
00203000
00204000
00205000
00206000
00207000
00208000
00209000
00210000
00211000
00212000

```

```

RUNKUT(N,X,XTL,FNMAX,TWOHC,TWOHCL,Y4,Y4L,Y1,Y1L,FH 00213000
INDX,*)FHL[INDX,*,F,G,GL]);
RUNKUT(N,X,XTL,FNMAX,HCHCL,Y4,Y4L,Y3,Y3L,FH[INDX,*) 00214000
FHL[INDX,*,F,G,GL]);
K:=K+1; 00215000
INDX:=(INDX+1)MOD CYM; 00216000
DOUBLE(K,O,H,HL,X,XL,+::=X,XTL); 00217000
F(N,X,XTL,Y3,Y3L,FH[INDX,*,FHL[INDX,*)]; 00218000
RUNKUT(N,X,XTL,FNMAX,HCHCL,Y3,Y3L,Y4,Y4L,FH[INDX,*) 00219000
FHL[INDX,*,F,G,GL]);
K:=K+1; 00220000
GO TO L5 00221000
END; 00222000
IF K=M THEN 00223000
BEGIN 00224000
INDX:=(INDX+1)MOD CYM; 00225000
DOUBLE(K,O,H,HL,X,XL,+::=X,XTL); 00226000
F(N,X,XTL,Y4,Y4L,FH[INDX,*,FHL[INDX,*)]; 00227000
FOR J:=1STEP 1UNTIL N DO 00228000
BEGIN 00229000
YFV[J]:=Y4[J]; 00230000
YFVL[J]:=Y4L[J] 00231000
END 00232000
END ELSE FOR J:=1STEP 1UNTIL N DO 00233000
BEGIN 00234000
YFV[J]:=Y3[J]; 00235000
YFVL[J]:=Y3L[J] 00236000
END 00237000
END ELSE GO TO L1 00238000
END ELSE 00239000
BEGIN 00240000
L6:IF CUMP(N,EAV,EKV,Y1,Y1L,Y4,Y4L)THEN 00241000
BEGIN 00242000
INDX:=(INDX+1)MOD CYM; 00243000
DOUBLE(K,O,H,HL,X,XL,+::=X,XTL); 00244000
F(N,X,XTL,Y4,Y4L,FH[INDX,*,FHL[INDX,*)]; 00245000
IF K<M THEN 00246000
BEGIN 00247000
IF K=L THEN FOR J:=1STEP 1UNTIL N DO 00248000
BEGIN 00249000
00250000
00251000
00252000

```

```

YFV[J]:=Y4[J];
YFVL[J]:=Y4L[J]
END;
RUNKUT(N,X,T,XTL,FNMAX,TWOHC,TWOHCL,Y4,Y4L,Y1,Y1L,FH
[INDX,*],FHL[INDX,*],F,G,GL);
RUNKUT(N,X,T,XTL,FNMAX,HC,HCL,Y4,Y4L,Y3,Y3L,FH[INDX,*],
FHL[INDX,*],F,G,GL);
K:=K+1;
INDX:=(INDX+1)MOD CYM;
DOUBLE(K,O,H,HL,X,XL,+,:=,XT,XTL);
F(N,X,T,XTL,Y3,Y3L,FH[INDX,*],FHL[INDX,*]);
IF K=L THEN FOR J:=1STEP 1UNTIL N DO
BEGIN
YFV[J]:=Y3[J];
YFVL[J]:=Y3L[J]
END;
RUNKUT(N,X,T,XTL,FNMAX,HC,HCL,Y3,Y3L,Y4,Y4L,FH[INDX,*],
FHL[INDX,*],F,G,GL);
K:=K+1;
GO TO L6
END;
FOR J:=1STEP 1UNTIL N DO
BEGIN
YIV[J]:=Y4[J];
YIVL[J]:=Y4L[J]
END
END ELSE GO TO L1
END
END;
DOUBLE(M,O,H,HL,X,XL,+,:=,X,XL)
END;
DSTART:=CNTR;
END;

```


III. THE EXECUTIVE PROCEDURE

A. Introduction

The executive procedure acts in an administrative and supervisory capacity. It does the bookkeeping and makes the decisions as to which methods are to be used, but does none of the actual integration. The executive procedure uses as subprocedures five basic integration routines; these are:

- a) The Adams-Bashforth-Moulton routine,
 - b) The Stetter-Gragg-Butcher routine,
 - c) The Cowell constant Nth order difference routine,
 - d) The Runge-Kutta-Shanks routine,
 - e) The start and restart routine
- (containing a separate Runge-Kutta-Shanks routine).

These five basic routines do the actual integration. A description of the single-precision subroutines is contained in [21] and will not be repeated here. Each of the double-precision routines is described in Chapter II of this report.

The executive procedure works in the following way. When a call is made in the procedure to integrate from point a to the point b, this interval is divided into eighths. The first eighth of the interval is integrated by one method for each of two different orders, and the time taken by each recorded. The second eighth is integrated by another method, also for two different orders, and the times recorded. The winners then compete against each other over the next fourth of the interval. That is, the faster order of the first

method and the faster order of the second method are both used to integrate the second fourth of the interval, and the time taken by each recorded. The faster method of these two is then presumably the best (fastest) of the four tried, and it is used alone to integrate over the last half of the interval. All of the times measured above are then logged in the cumulative history file of the appropriate problem type with the winners and losers noted. This file then is used as the basis of selecting which methods and orders are chosen each time, in such a way that the past performance of the different methods and orders influences the choice of which are allowed to compete.

B. The Selection Process

There are four methods available for the integration process, and within each method there are four orders available. Those for the single-precision program are as follows:

- a) Adams method with orders 5(4), 6(4), 7(4), 8(5),
- b) Butcher formulas with orders 5(4), 7(4), 9(4), 9(5),
- c) Cowell method with orders 7(5), 9(5), 11(5), 11(4),
- d) Shanks formulas with orders 4-4, 5-5, 6-7, 7-9.

Each order of each multistep method has an associated Runge-Kutta-Shanks restart procedure order given in parenthesis after the method order. Details on these methods are given in [21]. The disk file containing the coefficients has several additional orders of each method, but the single-precision program is now set to use just those mentioned above.

The methods and orders for the double-precision program are

Adams: 11(7), 12(7), 14(7), 15(7),
Butcher: 7(7), 9(7), 11(7), 13(8),
Cowell: 11(5), 13(7), 15(7), 17(7),
Shanks: 6-7, 7-9, 8-10, 8-12.

The selection process is the following. The first of two methods is chosen at random from among the three most successful available. The method showing the best history of success among the remaining methods is chosen as the second method, with the cumulative history file being used to determine the degree of success. Then within each method the same kind of selection process with respect to orders is used. That is, the first order is chosen at random, from the three most successful and the second order is chosen on the basis of which of the remaining has been the most successful (fastest running) order of that method. Thus it is seen that the past performance of the different methods and orders influences the choice of which are allowed to compete, such that the more successful have a higher probability of being selected.

In using the time as the sole estimate of performance efficiency, it is assumed that all orders and methods have satisfactorily met the accuracy requirements. The accuracy requirements of each method are met by controlling step size and making error estimates at each step. The method of error estimate is different for the different methods. In the Runge-Kutta-Shanks single step method, the error is estimated by taking two half steps and then a whole step. In the Adams and Butcher methods the difference between predictor and corrector is used. In the Cowell method a mid-range formula is used.

C. Organization of the History File

The history of the effectiveness of each method is recorded in a disk file with a separate history associated with each problem type. Problems are classified into twelve types based on three classifications.

The first classification is according to time taken for a function evaluation. This time is either long or short compared to the estimated time taken to evaluate the sums of products for a typical method; there are two divisions in this category.

The second classification is according to number of dependent variables; there are two divisions in this category. If the number of variables in the differential equations to be integrated is six or less, this is considered a "small" number of variables. If there are more than six variables, this is considered a "large" number of variables. This division is more or less arbitrary and could be changed easily.

The third classification is according to accuracy. There are three error ranges. For the single-precision program these are $> 10^{-3}$, 10^{-3} to 10^{-6} , $< 10^{-6}$; for double precision these are $> 10^{-8}$, 10^{-8} to 10^{-14} , $< 10^{-14}$. These correspond roughly to low, medium and high accuracies for the corresponding precision.

The two time division, two number of variable divisions, and three accuracy divisions form the twelve types. A complete and separate history file is kept for each of the problem types.

The histories are recorded in the following manner. Associated with each order of each method are two numbers. The first (a positive number) records the time associated with trials in which this order was the winner. The second (a negative number) records the time associated with trials

in which it was the loser. The sum of these two numbers is taken as the "score" or performance number and will be greater if the order of this method has been a consistent winner and will be less (more negative) if it has been a consistent loser.

Associated with each method then is a method score analogous to the order scores just described. That is, each method has one positive and one negative number recording the time spent winning and losing respectively. In addition to this, a history is also kept of which methods the wins and losses were against, but this part of the history is not used in selecting competitors.

The history file can be printed and punched by the program called WRITE HISTORY FILE. (This program is described in Chapter IV paragraph C.) In describing the output of this program, use will be made of an abbreviated notation. A stands for Adams method, B for Butcher, C for Cowell, and S for Shanks formulas. A number given following the letter designates the order of that method where 1 stands for the lowest order available, 2 for the next lowest, etc. Thus A3 stands for the second highest order Adams method. A sign following the letter or number designates winning time or losing time for this method-order. For example, B2⁺ designates winning time for Butcher's method, second lowest order; C⁻ designates losing time for Cowell's method; etc. Finally, if a letter follows the sign in parenthesis, this designates which method the win or loss was against; thus B⁺(A) designates winning time by Butcher against Adams. With this notation the organization of the history file is as follows:

The first three items (printed on the first line of the output of the history file) are not times but other bookkeeping items. The first number gives the date (in the form year, day) that this particular history file was initialized or last updated. The second number gives the total number of times the procedure has been called (using this particular history file). The third number gives the present value of the random number used in generating the random number sequence.

Following these three numbers come the cumulative times the various methods spent winning and losing. These are organized in a 9 row, 8 column matrix. The first 4 rows give wins and losses of the various orders within each method; that is, the results of the competitive trials over the $1/8$ sections of the range of integration. Table I gives this organization in terms of the notation described above.

Following this is a row giving cumulative winning and losing times by methods; that is, the results of the trials over the $1/4$ sections of the range of integration. This row is organized:

A^+ A^- B^+ B^- C^+ C^- S^+ S^- .

The last four rows give a more detailed breakdown of the line above, giving the method against which the winning and losing times were made. It is organized as the Table II. It is noted here that entries of the form $A^+(A)$, $B^-(B)$, $C^+(C)$, etc. will all be zero, since methods do not compete against themselves.

TABLE I
ORGANIZATION OF CUMULATIVE WINNING
AND LOSING TIMES BY METHOD AND ORDER

$A1^+$	$A1^-$	$A2^+$	$A2^-$	$A3^+$	$A3^-$	$A4^+$	$A4^-$
$B1^+$	$B1^-$	$B2^+$	$B2^-$	$B3^+$	$B3^-$	$B4^+$	$B4^-$
$C1^+$	$C1^-$	$C2^+$	$C2^-$	$C3^+$	$C3^-$	$C4^+$	$C4^-$
$S1^+$	$S1^-$	$S2^+$	$S2^-$	$S3^+$	$S3^-$	$S4^+$	$S4^-$

Notation here: A = Adams, B = Butcher, C = Cowell, S = Shanks;
1 = lowest order, 2 = second lowest order, etc;
+ stands for win, - stands for loss.

TABLE II
ORGANIZATION OF CUMULATIVE WINNING
AND LOSING TIMES BY METHOD VS. METHOD

$A^+(A)$	$A^-(A)$	$B^+(A)$	$B^-(A)$	$C^+(A)$	$C^-(A)$	$S^+(A)$	$S^-(A)$
$A^+(B)$	$A^-(B)$	$B^+(B)$	$B^-(B)$	$C^+(B)$	$C^-(B)$	$S^+(B)$	$S^-(B)$
$A^+(C)$	$A^-(C)$	$B^+(C)$	$B^-(C)$	$C^+(C)$	$C^-(C)$	$S^+(C)$	$S^-(C)$
$A^+(S)$	$A^-(S)$	$B^+(S)$	$B^-(S)$	$C^+(S)$	$C^-(S)$	$S^+(S)$	$S^-(S)$

Notation here: A = Adams, B = Butcher, C = Cowell, S = Shanks;
+ stands for win, - stands for loss.
 $A^+(S)$ stands for Adams win against Shanks,
 $C^-(B)$ stands for Cowell loss against Butcher, etc.
Entries of the form $A^+(A)$, or $C^-(C)$ etc., should all be zero since a
method does not compete against itself.

D. Inputs to the Executive Procedure

1. The Single-Precision Program

A call on the single-precision executive procedure would look like the following:

```
DIFEQINT (N, XI, XF, Y, F, P, EA, ER, DX)
```

Here the identifiers in parenthesis are the inputs to the procedure and represent the following information:

N is the number of equations in the system to be integrated,

XI is the initial value of the independent variable,

XF is the final value of the independent variable,

Y is the initial value of the dependent variable. Y is a vector (one-dimensional array). At the conclusion of the procedure, Y is set to the final values of the dependent variable; that is, Y is also the output variable.

F is the procedure for calculating dy/dx as a function of x and y . This procedure must be written by the user and describes the system of differential equations being integrated. It must be written so as to have four parameters:

- a) N, the number of equations,
- b) X, the independent variable,
- c) Y, the dependent variable (vector),
- d) FV, the vector values of dy/dx at the point x, y .

The first three parameters are input and FV is the output.

P is the error accumulation parameter. This parameter expresses

the user's opinion as to how the errors are going to accumulate over the range of integration. For example, if it is expected that the errors will be random then P would be set to 0.5. If it is expected that the errors will accumulate linearly then set P to 1. These are the two most usual cases but other situations can occur.

EA is the absolute error vector. This vector gives the acceptable absolute errors in the value of Y final.

ER is the relative error vector. This vector gives the acceptable relative error in the value of Y final. It is the weaker of the two conditions EA and ER that is met for each component of the vector Y.

DX is the estimated value of the initial step size. This estimate need not be especially accurate since the individual methods will adjust the step size to the appropriate value.

2. The Double-Precision Program

A call on the double-precision procedure would look like the following:

DIFEQINT (N, XI, LXI, XF, LXF, Y, LY, F, P, EA, ER, DX)

Here the identifiers in the parenthesis are the inputs to the procedure and represent the following information:

N is the number of equations in the system being integrated.

XI and LXI are high part and low part of the double-precision initial values of the independent variables.

XF and LXF are the high and low parts of the final values of the independent variable.

Y and LY are the high and low parts of the initial value of the dependent variable vector. This is also the output vector.

F is a procedure for calculating dy/dx as a function of x and y. This procedure must be written by the user and describes the system of differential equations being integrated. It must be written so as to have the following parameters:

- a) N, the number of variables,
- b) X and LX, the high and low parts of the independent variables,
- c) Y and LY, the high and low parts of the dependent variable (vector),
- d) FV and LFV, the high and low parts of dy/dx at the point x, y.

A call on the procedure F would look like F (N, X, LX, Y, LY, FV, LFV).

The first five parameters are input and the last two are output.

P, EA, ER, and DX are error accumulation parameter, absolute error vector, relative error vector, and suggested starting step size. These parameters are given in single precision only and are identical to the corresponding parameters in the single precision version (see page 107).

E. Updating of the History File and Forgetting

The times recorded in the history file are cumulative. That is, after a competition is held, the times taken by the competing methods and orders are added to (for a win) or subtracted from (for a loss) the appropriate positions in the history file. Thus, the entries in the history tables represent an index expressing the cumulative past performance.

Decisions as to which method or order within a method is considered to have the best performance history are made on the basis of the sum of the win and loss entries for that method or order. The method or order having the maximum value for this sum is considered to have best history (remembering that the loss entries are negative). One notes that not all the history file is used in the decision making process; in particular, those entries in Table II are not used in decision making but are recorded only to give the user a more detailed account of the competitions.

One further feature is introduced into the learning process and this is the gradual "forgetting" of events in the more distant past. This is accomplished by multiplying those history scores used in the decision making by a factor less than one, just before the most recent histories are added. This causes the events in the distant past to have less influence than those more recent in determining the performance figure of an order and method. The factor used is 0.98 but it is not known what would be the optimum factor. Note that the entries in the history file described in Table II do not involve forgetting. Since these entries are not used in any decision making process but only tabulated for the user's interest, forgetting would serve no practical purpose here. The entries in Table II thus represent a total or unattenuated history of the competition between the various methods.

F. Reading of Coefficients and History Files

Also needed as input to the executive routine are the tables of coefficients associated with the various methods and the past performance

history files. The coefficients are read in by the procedure the first time the procedure is called and a flag set to indicate that these have been read in once; they are stored in an array declared OWN and need not be read in again during the operation of the program.

The coefficients are stored on a disk file called "TAPE831". It contains the following coefficients:

- a) Adams' method, orders 4 through 19,
- b) Butcher's formulas, orders 3, 5, 7, 9, 11, 13,
- c) Cowell's method, orders 7, 9, 11, 13, 15, 17, 19,
- d) Shanks formulas, orders 4(4), 5(5), 6(7), 7(7), 7(9), 8(10), 8(12).

Only four orders of each method are actually used in each program.

The histories for the single-precision program are stored in a disk file called "REMOTE" "A831S**", and those of the double-precision program are stored in a file called "REMOTE" "A831**D". Each time the executive procedure is called the history associated with the appropriate problem type is read in from the disk.

G. Outputs of the Executive Procedure

The executive procedure returns the final value of the dependent variable as its principal output. This is returned through the same variable, the vector Y (or Y and LY in double) described in inputs to the procedure, paragraph D of this chapter.

There are several other types of output that are printed. Printed in the file called "HISTORY" is a pair of numbers giving the method and order that is about to be used and the times for each order and method

after the comparisons have been made. This information is printed in a two digit code, the first digit representing the method and the second (if present) indicating the order. The method code is:

0 represents Adams,
1 represents Butcher,
2 represents Cowell,
3 represents Shanks.

The order code is such that 0 represents the lowest order, 1 represents the next lowest order, etc.

Also printed in the file "HISTORY" are the results of comparison runs in which the results (values of the dependent variable) of the two competing method orders differ by more than twice the allowed errors. Also printed are the initial and final values of the independent variable, the two differing values of the dependent variables, and an integer telling which component of the dependent variable appears to be in error.

Other messages associated with anomolous conditions are also printed in this file. In particular an integer overflow condition occurs if the step size collapses. Recovery from step size collapse can usually be effected, but the message "INTEGER OVERFLOW" will be printed in file "HISTORY" whenever it occurs.

Finally, the procedure outputs the updated performance history by writing it back into the appropriate file on the disk.

H. Flow Diagram and Program Listing

Figure 6 is the flow diagram for the executive procedure. (The flow diagrams for the single and double precision are essentially the same).

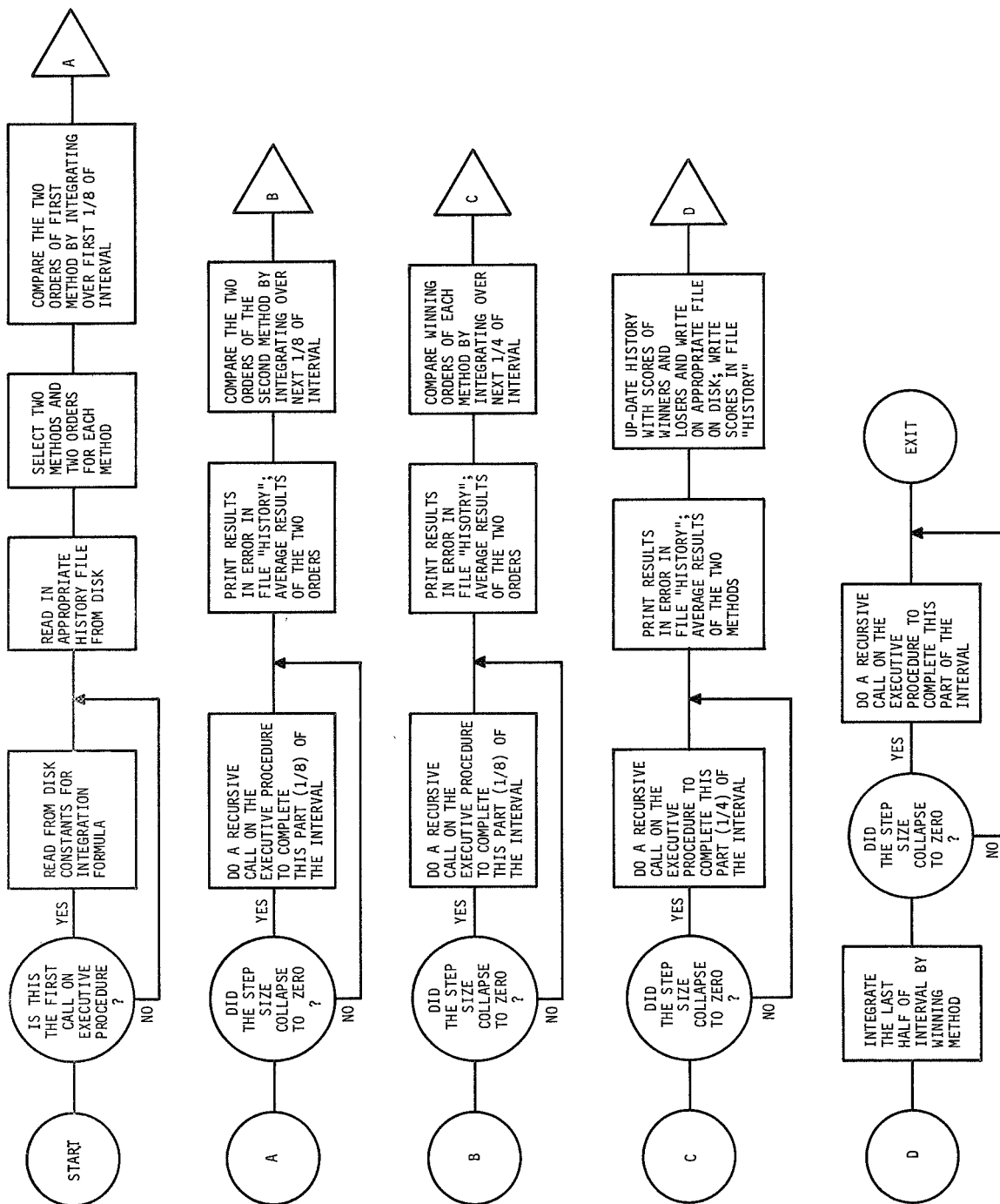


Figure 6. Flow Diagram for the Executive Procedure.

The program listings for both the single and double-precision version of the executive procedure follow at the end of this section. The listings of the individual methods and restart programs are given in their "squeezed" form. The double-precision versions of the individual methods are also given (unsqueezed) at the end of their respective sections in this report. The squeezed Adams procedure listed here is not an exact copy of the unsqueezed version listed at the end of section A, chapter II but is equivalent to the UEB = FALSE iteration control option described there. The single-precision individual methods are also given (unsqueezed) in reference [21].

00003000
00004000
00005000
00006000
00007000
00008000
00009000
00010000
00011000
00012000
00013000
00014000
00015000

```

FILE PUN 15 "HISTORY"(3,15);
PROCEDURE DIFEQINT(N,XI,XF,LXI,XF,LXF,Y,LX,F,P,EA,ER,DX);
VALUE N,XF,XI,DX,P,LXI,LXF;
INTEGER N;
REAL XI,XF,DX,P,LXI,LXF;
PROCEDURE F;
ARRAY Y,LX,EA,ER[0];

BEGIN
  REAL TEMP,LTEMP;
  DOUBLE(XI,LXI,XF,LXF,TEMP,LTEMP);
  IF TEMP # 0 THEN
    BEGIN

```

```

      INTEGR
      4
      5
      6
      7
      8
      9
      10
      11
      12
      13
      14
      15
      16
      17
      18
      19
      20
      21
      22
      23
      24
      25
      26
      27
      28

ER PROCEDURE DSTART(N,XI,XF,XFL,C1,EA,ER,F,M,X,XL,YIV,YIVL,YH,YHL,FH
,FHL,YFV,YFVL,CYI,CYM,PA,P,FNEVAL,RKSCONST,RKSCONSTL);VALUE N,XI,XIL,XF,
XFL,C1,M,CYI,CYM,PA,P,FNEVAL;INTEGER N,C1,M,CYI,CYM,PA,FNEVAL;REAL XI,XI
L,XF,XFL,X,XL,P;REAL ARRAY EA,ER,YIVL,YFV,YFVL,RKSCONST,RKSCONSTL[0]
,YH,YHL,FH,FHL[0,0];PROCEDURE F;BEGIN INTEGER I,J,K,L,COEFFCNT,FNMAX,IND
X,NINDX,CNTR;REAL INT,INTL,H,HL,TWOHL,T1,T1L,XT,XTL;REAL ARRAY HC,H
CL,TWOHC,TWOHCL[0:((FNEVAL+3)*FNEVAL)/2)-2],EAV,ERV,Y1,Y1L,Y2,Y2L,Y3,Y3
L,Y4,Y4L[0:N],G,GL[0:FNEVAL-2,0:N];LABEL L1,L2,L3,L4,L5,L6;PROCEDURE RUN
KUT(N,X,XL,FNMAX,COEFF,COEFFL,YIV,YIVL,YFV,YFVL,FV,FVL,F,G,GL);VALUE N,X
XL,FNMAX;INTEGER N,FNMAX;REAL X,XL;REAL ARRAY COEFF,COEFFL,YIV,YIVL,YFV
,YFVL,FV,FVL[0],G,GL[0,0];PROCEDURE F;BEGIN INTEGER I,J,K,CNTR;REAL TEMP
,TEMPL;TEMP:=COEFF[CNT:0];TEMPL:=COEFFL[CNT];FOR I:=0STEP 1UNTIL FNMA
X DO BEGIN FOR J:=1STEP 1UNTIL N DO DOUBLE(FV[J],FVL[J],TEMP,TEMPL,X,YIV
[J],YIVL[J],+,:=, YFV[J],YFVL[J]);FOR K:=0STEP 1UNTIL I-1 DO BEGIN TEMP:=C
OEFF[CNT:+=CNTR+1];TEMPL:=COEFFL[CNT];FOR J:=1STEP 1UNTIL N DO DOUBLE(G
[K,J],GL[K,J],TEMP,TEMPL,X,YFV[J],YFVL[J],+,:=, YFV[J],YFVL[J]);END;CNTR:=
CNTR+1;DOUBLE(X,XL,COEFF[CNT],COEFFL[CNT],+,:=,TEMP,TEMPL);F(N,TEMP,TE
MPL,YFV,YFVL,G[I,*],GL[I,*]);TEMP:=COEFF[CNT:+=CNTR+1];TEMPL:=COEFFL[CNT
];END;FOR J:=1STEP 1UNTIL N DO DOUBLE(FV[J],FVL[J],TEMP,TEMPL,X,YIV[J],Y
IVL[J],+,:=, YFV[J],YFVL[J]);FOR K:=0STEP 1UNTIL FNMAX DO BEGIN TEMP:=COE
FF[CNT:+=CNTR+1];TEMPL:=COEFFL[CNT];FOR J:=1STEP 1UNTIL N DO DOUBLE(G[K
,J],GL[K,J],TEMP,TEMPL,X,YFV[J],YFVL[J],+,:=, YFV[J],YFVL[J]);END;END;BOOL
EAN PROCEDURE COMP(N,EAV,ERV,YL,Z,ZL);VALUE N;INTEGER N;REAL ARRAY EAV
,ERV,Y,YL,Z,ZL[0];BEGIN INTEGER J;REAL T1,T1L;LABEL L1;FOR J:=1STEP 1UNT

```

```

29 IL N DO BEGIN DOUBLE(Y[J],YL[J],Z[J],ZL[J],,,T1,T1L);IF(T1:=ABS(T1))>
30 EAV[J]THEN BEGIN IF T1>ERV[J]*ABS(Z[J])THEN BEGIN COMP:=FALSE;GO TO L1 E
31 ND END;COMP:=TRUE;L1:=END;CNTR:=1;IF M#0THEN BEGIN COEFFCNT:=((CFNEVA
32 L+3)*FNEVAL)/2)-2;FNMAX:=FNEVAL-2;DOUBLE(XF,XFL,XI,XIL,,:=INT,INTL);DO
33 UBLE(INT,INTL,INT,INTL,+,C1,0,/,:=,TWOH,TWOHL);DOUBLE(INT,INTL,C1,0,/,:=
34 ,H,HL);FOR I:=0STEP 1UNTIL COEFFCNT DO BEGIN T1:=RKSCONST[I];T1L:=RKSCON
35 STL[I];DOUBLE(C1,T1L,H,HL,x,:=,HCL[I],HCL[I]);DOUBLE(T1,T1L,TWOH,TWOHL,x,
36 :=,TWOHCL[I],TWOHCL[I]);END;INDEX:=CYI MOD CYM;T1:=(C1/2)*P;FOR J:=1STEP 1U
37 NTIL N DO BEGIN EAV[J]:=EAV[J]/T1;ERV[J]:=ERV[J]/T1 END;IF PA=2THEN BEGIN
38 NINDEX:=(INDEX+1)MOD CYM;K:=0;XTL:=X;GO TO L3 END;IF PA=1THEN L:=M
39 DIV 2;RUNKUT(N,X,XL,FNMAX,TWOHCL,TWOHCL,YIV,YIVL,Y1,Y1L,FHLIN,X,*,J,FHLIN
40 DX,*,J,F,G,GL);GO TO L4;L1:=TWOH:=H;TWOHL:=HL;CNTR:=CNTR+CNTR;C1:=C1+C1;DO
41 UBLE(INT,INTL,C1,0,/,:=,H,HL);FOR I:=0STEP 1UNTIL COEFFCNT DO BEGIN TWOH
42 CL[I]:=HCL[I];TWOHCL[I]:=HCL[I];DOUBLE(RKSCONSTL[I],RKSCONSTL[I],H,HL,x,*,J,
43 HCL[I],HCL[I]);END;INDEX:=CYI MOD CYM;T1:=(C1/2)*P;IF PA=2THEN BEGIN K:=0;N
44 INDEX:=(INDEX+1)MOD CYM;FOR J:=1STEP 1UNTIL N DO BEGIN EAV[J]:=EAV[J]/T1;ER
45 V[J]:=ERV[J]/T1;Y1L:=YHININDEX,J;Y1L:=YHININDEX,J;END;L2:=DOUBLE(K,0,
46 H,HL,x,*,J,XL,+,:=,XT,XTL);RUNKUT(N,XT,XTL,FNMAX,HC,HCL,YHININDEX,*,J,YHLEIND
47 X,*,J,YHLEINDEX,*,J,YHLEINDEX,*,J,FHLIN,X,*,J,F,G,GL);K:=K+1;INDEX
48 :=NINDEX;NINDEX:=(INDEX+1)MOD CYM;DOUBLE(K,0,H,HL,x,*,J,XL,+,:=,XT,XTL);F(N,X
49 T,XTL,YHININDEX,*,J,YHLEINDEX,*,J,FHLIN,X,*,J);RUNKUT(N,XT,XTL,FNM
50 AX,HC,HCL,YHININDEX,*,J,YHLEINDEX,*,J,YHLEINDEX,*,J,FHLIN,X,*,J,FHL
51 INDEX,*,J,F,G,GL);IF COMP(N,EAV,ERV,Y1,Y1L,YHININDEX,*,J,YHLEINDEX,*,J)THEN
52 BEGIN K:=K+1;IF K<M THEN BEGIN INDEX:=NINDEX;NINDEX:=(INDEX+1)MOD CYM;DOUBLE
53 (K,0,H,HL,x,*,J,XL,+,:=,XT,XTL);F(N,XT,XTL,YHININDEX,*,J,YHLEINDEX,*,
54 J,FHLIN,X,*,J);L3:=RUNKUT(N,XT,XTL,FNMAX,TWOHCL,TWOHCL,YHININDEX,*,J,YHLEIND
55 X,*,J,Y1,Y1L,FHLIN,X,*,J,FHLIN,X,*,J,F,G,GL);GO TO L2 END;IF K=M THEN BEGI
56 N INDEX:=NINDEX;DOUBLE(K,0,H,HL,x,*,J,XL,+,:=,XT,XTL);F(N,XT,XTL,YHININDEX,*,J,
57 YHLEINDEX,*,J,FHLIN,X,*,J,FHLIN,X,*,J);END END ELSE GO TO L1 END ELSE BEGIN
58 FUR J:=1STEP 1UNTIL N DO BEGIN EAV[J]:=EAV[J]/T1;ERV[J]:=ERV[J]/T1;Y1L:=
59 Y2[J];Y1L:=Y2L[J];END;L4:=RUNKUT(N,X,XL,FNMAX,HC,HCL,YIV,YIVL,Y2,Y2L,FH
60 LINDEX,*,J,FHLIN,X,*,J,F,G,GL);INDEX:=(INDEX+1)MOD CYM;DOUBLE(X,XL,H,HL,+,:=
61 XT,XTL);F(N,XT,XTL,Y2,Y2L,FHLIN,X,*,J,FHLIN,X,*,J);RUNKUT(N,XT,XTL,FNMAX
62 ,HC,HCL,Y2,Y2L,Y4,Y4L,FHLIN,X,*,J,FHLIN,X,*,J,F,G,GL);K:=2;IF PA=0THEN BE
63 GIN L5:=IF COMP(N,EAV,ERV,Y1,Y1L,Y4,Y4L)THEN BEGIN IF K<M THEN BEGIN INDEX
64 :=(INDEX+1)MOD CYM;DOUBLE(K,0,H,HL,x,*,J,XL,+,:=,XT,XTL);F(N,XT,XTL,Y4,Y4L,
65 FHLIN,X,*,J,FHLIN,X,*,J);RUNKUT(N,XT,XTL,FNMAX,TWOHCL,TWOHCL,Y4,Y4L,Y1,Y1L
66 ,FHLIN,X,*,J,FHLIN,X,*,J,F,G,GL);RUNKUT(N,XT,XTL,FNMAX,HC,HCL,Y4,Y4L,Y3,Y
67 3L,FHLIN,X,*,J,FHLIN,X,*,J,F,G,GL);K:=K+1;INDEX:=(INDEX+1)MOD CYM;DOUBLE(K,
68 0,H,HL,x,*,J,XL,+,:=,XT,XTL);F(N,XT,XTL,Y3,Y3L,FHLIN,X,*,J,FHLIN,X,*,J);RUN

```



```

109 BL[MU]]:DOUBLE(ADAMSCOEFF[MU+Q],ADAMSCOEFFL[MU+Q],H,HL,X,*,HRS[MU],HBSL[
110 MU])$END$DOUBLE(BSQZ,BSQZL,H,HL,X,*,HBSQZ,HBSQZL)$CU+(C1*(-P))$XGR$ALLI 8
111 EGIN LEAII+((UEAII+EAII+XCU)*C2MQP5)LERI]+(UERII+ERII+XCU)*C2MQP5$END
112 $PREDICT:DOUBLE(C2,C2L,1.0,*,*,C2,C2L)$DOUBLE(XF,XFL,H,HL,C2,C2L,X,*,*,
113 X,XL)$CYCMU+(CYCMU+QMINUS1)$MOD QT2M1$ALLMU BEGIN CYCMU+(CYCMU+1)$MOD QT2M
114 $DOUBLE(HB[MU],HBL[MU],*,HBMU,HBMUL)$DOUBLE(HBS[MU],HBSL[MU],*,HBSMU,HBS
115 MUL)$ALLI BEGIN DOUBLE(FMUI+FHICYCMU,II,FMUIL+FHLCYCMU,II,HBMU,HBMUL,X
116 YPII,YPLII,*,*,YPII,YPLII)$DOUBLE(CII,CLII)$FMUI,FMUIL,HBSMU,HBSMU
117 L,X,*,*,CII,CLII)$END$END$F(N,X,XL,YP,YPL,FNU,FNU)$CORRECT$ALLI BEGIN
118 DOUBLE(FNUI+FNULI,II,FNUI,II,HBSQZ,HBSQZL,X,CII,CLII,*,*,YCY,YCIL)$DOUB
119 LE(YCII,YCII)$YCI+YCI$YCI+YCI$CHANGE$CHANGE$YF CHANGE+ABS(
120 CHANGE)$UEAII$THEN IF CHANGE>ABS(UERI+XCYI)$THEN BEGIN WHILE I<N DO DOUB
121 LE(FNUI+I+1)$FNULI,II,HBSQZ,HBSQZL,X,CII,CLII,*,*,YCY,YCIL)$F(N,X,
122 XL,YC,YCL,FNU,FNU)$GO TO CORRECT$END$TEST$F(N,X,XL,YC,YCL,FNU,FNU)$
123 $TOOSMALL+TRUE$ALLI BEGIN DOUBLE(FNUI,II,FNUI,II,HBSQZ,HBSQZL,X,CII,CLII
124 I,*,*,YCY,YCII)$DOUBLE(YCI+YCI,II,YCII,II,YPII,YPLII,*,*,ERROR,ERRD
125 RL)$IF ERROR+ABS(ERROR)$UEAII$THEN IF ERROR>ABS(UERI+XCYI)$THEN GO TO HA
126 LF$IF TOOSMALL THEN IF ERROR>LEAII$THEN IF ERROR>ABS(LERI+XCYI)$THEN GO
127 SMALL+FALSE$END$ACCEPT$POINTCOUNT+POINTCOUNT+1$CYCMU+(CYCMU+1)$MOD QT2M1$
128 ALLI BEGIN DOUBLE(FNUI,II,FNUI,II,FHICYCMU,II,FHLCYCMU,II)$DOUBLE(CII
129 +YPII+YCI,II,YPLII+YCLII,II,YPII,YPLII)$END$IF C2<1.0 THEN GO TO F
130 INISH$IF TOOSMALL THEN BEGIN IF SMALLCOUNT+SMALLCOUNT+1>23 THEN IF POINTCO
131 UNT2QT2M1 THEN IF C22.0 THEN GO TO DOUBLE$END ELSE SMALLCOUNT+0$GO TO PR
132 EDICT$DOUBBLE(K+J+CYCMU)$FOR MU+1 STEP 1 UNTIL QMINUS1 DO BEGIN IF J+J-1<0 TH
133 EN J+QT2M2$IF K+K-2<0 THEN K+K+QT2M1$ALLI DOUBLE(FHCK,II,FHLC,II,*,*,FHIJ,
134 II,FHLL,II)$END$C1+C1 DIV 2$DOUBLE(C2,C2L,2.0,*,*,C2,C2L)$GO TO RESET$
135 HALF$DOUBLE(C2,C2L,1.0,*,*,C2,C2L)$DOUBLE(XF,XFL,H,HL,C2,C2L,X,*,*,X,XL
136 )$C1+C1+1$DOUBLE(C2,C2L,C2L,*,*,C2,C2L)$IF C22Q THEN GO TO RESTART$F
137 INISH$IF C2=0 THEN GO TO LEAVE$D$SHANKS(N,X,XL,XF,XFL,Y,YL,F,RKSFNEVAL,RK
138 SURDER,RKSCOEFF,RKSCOEFFL,P,EA,ER,XF,X)$LEAVE$END$PROCEDURE BUTCHER(N,XI
139 ,XIL,XF,XFL,Y,YL,F,EA,ER,P,D,X,RKSFN,RKSDORDER,RKSCOEFF,RKSCOEFFL,B00GERFA
140 CTOR,K,BUTCHERCOEFF,BUTCHERCOEFFL,START,D$SHANKS)$VALUE N,XI,XIL,XF,XFL,P
141 ,DX,RKSFN,RKSDORDER,K,$INTEGER N,RKSFN,RKSDORDER,K,$REAL XI,XIL,B00GERFACT
142 OR,XF,XFL,P,DX,$REAL ARRAY Y,YL,EA,ER,RKSCOEFF,RKSCOEFFL,BUTCHERCOEFF,BU
143 TCHERCOEFFL,IO)$PROCEDURE F,D$SHANKS,$INTEGER PROCEDURE START,$BEGIN INTEG
144 ER I,J,C1,KM1,TKM1,INDX,II,I2,CYI,DELAYCNT,HISTORYCNT,YKM2,$REAL INT,INT
145 L,C2,C2L,DFACTOR,X,XL,H,HL,T1,T1L,T2,T2L,T3,T3L,T4,T4L,T5,T5L,T6,T6L,T7,
146 T7L,T8,T8L,THETA,THETAH,THETAHL,HBL,HBEI,A,HBETAL,HBETA0,HBETA0,HBETA0
147 L,X,T,XTL,$REAL ARRAY YH,YHL,FH,FHLL,IO,K+K-2,0,NJ,YT,YTL,YP,YPL,YC,YCL,F
148 T,FTL,EA,ERV,EAVD,ERVD,IO,NJ,YCOEFF,YCOEFFL,PCOEFF,PCOEFFL,CCOEFF,CCOEFF

```



```

189 ,YTL[J],DOUBLE(T7,T7L,T3,T3L,x,T8,T8L,T4,T4L,x+,YPL[J],YPL[J],+,+,YPL[J]
190 ,YPL[J])DOUBLE(T7,T7L,T5,T5L,x,T8,T8L,T6,T6L,x+,YCL[J],YCL[J],+,+,YCL[
191 J],YCL[J])END END ,F(N,X,T,XTL,YT,YTL,FT,FTL),FOR J :=1 STEP 1 UNTIL N DO
192 BEGIN T1 :=FTL[J],T1L :=FTL[J],DOUBLE(CHB,HBL,T1,T1L,x,YPL[J],YPL[J],+,+,
193 YPL[J],YPL[J])DOUBLE(CHBETA,HBETA,T1,T1L,x,YCL[J],YCL[J],+,+,YCL[J],YCL[J]
194 )END ,F(N,X,XL,Y,YPL,FT,FTL),INDEX :=(INDEX +1)MOD TKM1 ,FOR J :=1 STEP
195 1 UNTIL N DO BEGIN DOUBLE(CHBETAO,HBETAO,FTL[J],FTL[J],x,YCL[J],YCL[J],+,+,
196 ,T3,T3L),DOUBLE(T3,T3L,YPL[J],YPL[J],+,+,T2,T2L),IF (T2 :=ABS(T2))>EAVD
197 {J}THEN BEGIN IF T2 >ERVD{J}*ABS(T3)THEN BEGIN IF T2 >EAV{J}THEN BEGIN I
198 F T2 >ERV{J}*ABS(T3)THEN GO TO L2 END ,YH{INDEX,J}:=T3 ,YH{INDEX,J}:=T3L
199 ,FDR J :=J +1 STEP 1 UNTIL N DO BEGIN DOUBLE(CHBETAO,HBETAO,FTL[J],FTL[J]
200 ,x,YCL[J],YCL[J],+,+,T3,T3L),DOUBLE(T3,T3L,YPL[J],YPL[J],+,+,T2,T2L),IF
201 (T2 :=ABS(T2))>EAV{J}THEN BEGIN IF T2 >ERV{J}*ABS(T3)THEN BEGIN L2:C1 :=
202 C1 +C1 ,DOUBLE(XF,XFL,C2,C2L,H,HL,x,+,+,X,XL),DOUBLE(C2,C2L,C2,C2L,+,+,
203 ,C2,C2L),INDEX :=(INDEX +TKM2)MOD TKM1 ,IF C2 <KM1 THEN GO TO CLOSER ,GO T
204 O STARTER END END ,YH{INDEX,J}:=T3 ,YH{INDEX,J}:=T3L END ,DELAYCNT :=0 ,F
205 (N,X,XL,YH{INDEX,J},YH{INDEX,J},FHL{INDEX,J},FHL{INDEX,J}),GO TO L3 END END
206 ,YH{INDEX,J}:=T3 ,YH{INDEX,J}:=T3L END ,F(N,X,XL,YH{INDEX,J},YH{INDEX,J},F
207 HL{INDEX,J},FHL{INDEX,J}),IF DELAYCNT >2 THEN BEGIN IF HISTORCNT >TKM1 THE
208 N BEGIN DOUBLE(C2,C2L,1,0,+,+,2,0,+,+,C2,C2L),IF C2 <1 THEN GO TO CLOSE
209 R ,C1 :=C1 DIV 2 ,CYI :=INDEX +TKM1 ,FOR I :=1 STEP 1 UNTIL KM1 DO BEGIN
210 I1 :=(I2 :=CYI -1)MOD TKM1 ,I2 :=(I2 -1)MOD TKM1 ,FOR J :=1 STEP 1 UNTIL
211 N DO BEGIN YH{I1,J}:=YH{I2,J},YH{I1,J}:=YH{I2,J},FHL{I1,J}:=FHL{I2,J},F
212 HL{I1,J}:=FHL{I2,J}END END ,GO TO DOUBLER END END ,DELAYCNT :=DELAYCNT +
213 1 ,L3:DOUBLE(C2,C2L,1,0,+,+,C2,C2L),IF C2 >1 THEN BEGIN HISTORCNT :=H
214 ISTORCNT +1 ,GO TO ACCEPT END ,CLOSER:IF C2 >0 THEN DSHANKS(N,X,XL,XF,x
215 FL,YH{INDEX,J},YH{INDEX,J},F,RKSN,RKSN,RKSN,RKSCOFF,RKSCOFFL,P,EA,ER,AB
216 S(INT)/C1),FOR J :=1 STEP 1 UNTIL N DO BEGIN YC{J}:=YH{INDEX,J},YCL{J}:=YH{
217 INDEX,J}END ,JEND ,PROCEDURE COWELL(N,XI,XIL,XF,XFL,Y,YL,F,EA,ER,P,DX,RKS
218 FN,RKSN,RKSN,RKSCOFF,RKSCOFFL,COWELLCOEFF,COWELLCOEFFL,START,DSHANKS)
219 ,VALUE N,XI,XIL,XF,XFL,P,DX,RKSN,RKSN,RKSN,RKSN,RKSN,RKSN,RKSN,RKSN,RKSN,RKSN
220 ,REAL XI,XIL,XF,XFL,P,DX ,REAL ARRAY Y,YL,EA,ER,RKSCOFF,RKSCOFFL,COWE
221 LCOEFF,COWELLCOEFFL),PROCEDURE F,DSHANKS ,INTEGER PROCEDURE START ,BE
222 GIN INTEGER C1,M,MW1,QM1,QP1,TQP1,INDEX,I1,I2,I3,I4,I,J,K,CYI ,INTEGER CO
223 RRECTCNT ,REAL INT,INTL,C2,C2L,DFACTOR,X,XL,H,HL,T1,T1L,T2,T2L,T3,T3L,T4
224 ,T4L,TS,T5L,T6,T6L ,BOOLEAN DFLAG,PFLAG ,REAL ARRAY FH,FHL{0:0 +Q,0:NI},Y
225 MID1,YMID1L,Y,YPL,YC,YCL,YM,YML,CS,CSL,FP,FPL,HDMIF,HDMIFL,HDMIFMID,HDM
226 IFMIDL,EAV,ERV,EAVD,ERVD{0:NI},PCOEFF,PCOEFFL,CCOEFF,CCOEFFL,MCOEFF,MCOEF
227 F{0:Q},LABEL L0,L1,L2,L3,L4,L5,STARTER,DOUBLER,ACCEPT,CORRECT,CLOSER,DO
228 UBLE(XF,XFL,XI,XIL,+,+,INT,INTL),C1 :=1 ,L0:C1 :=C1 +C1 ,IF C1 <0 THEN

```

```

229 GU TO LO ;IF DX <0 THEN DX :=-DX ;IF DX #0 THEN BEGIN L1:=IF ABS(INT)/C1
230 >DX THEN BEGIN C1 :=C1 +C1 ;GO TO L1 END END ;DOUBLE(C1,0,1,0,x,1:=C2,C
231 2L);M :=Q DIV 2 ;MM1 :=M -1 ;QM1 :=Q -1 ;QPI :=Q +1 ;TQPI :=QPI +Q ;DFAC
232 TOR :=2.0 *(Q +3);INDX :=0 ;X :=XI ;XL :=XIL ;F(N,X,YL,FH[0,*,J],FHL[
233 0,*,J]);STARTER:C1 :=(I1 :=START(N,XI,XIL,XF,XFL,C1,EA,ER,F,Q,X,XL,Y,YL,FH
234 ,FHL,FH,YMID1,YMID1L,INDX,TQPI,1,P,RKSFN,RKSCOEFF,RKSCOEFFL))X C1 ;DO
235 UBLE(C2,C2L,I1,0,x,Q,0,*,*,C2,C2L);INDX :=(INDX +Q)MOD TQPI ;IF C2 <M T
236 HEN GO TO CLOSER ;DBUBLER:DOUBLE(INDX,INTL,C1,0,*,*,H,HL);FOR K :=0 STEP
237 1 UNTIL Q DO BEGIN DOUBLE(H,HL,COWELLCOEFF[K],COWELLCOEFFL[K],*,*,PCOE
238 FF[K],PCOEFFL[K]);I1 :=K +QPI ;DOUBLE(H,HL,COWELLCOEFFL[K],*,*,PCOE
239 I1),*,*,CCOEFF[K],CCOEFFL[K]);I1 :=I1 +QPI ;DOUBLE(H,HL,COWELLCOEFFL[
240 I1],*,*,CCOEFFL[K],CCOEFFL[K]);END ;T1 :=C1 *P ;FOR J :=1 S
241 TEP 1 UNTIL N DO BEGIN EAVD[J]:=(EAV[J]:=EAL[J])/T1)/DFACTOR ;ERVVD[J]:=(ER
242 VJ):=ER[J]/T1)/DFACTOR END ;T1 :=MCOEFFL[0];T1L :=MCOEFFL[0];FOR J :=1 S
243 TEP 1 UNTIL N DO DOUBLE(YMID1[J],YMID1L[J],T1,T1L,FH[INDX,J],FHL[INDX,J]
244 ,*,*,HDM1F[J],HDM1FL[J]);CYI :=INDX +TQPI ;I3 :=CYI -QPI ;FOR K :=1 S
245 TEP 1 UNTIL M DO BEGIN I1 :=(CYI -K)MOD TQPI ;I2 :=(I3 +K)MOD TQPI ;DOUB
246 LE(MCOEFFL[K],MCOEFFL[K],H,HL,*,*,T1,T1L);T2 :=MCOEFFL[I4 :=QPI -K];T2L :=
247 MCOEFFL[I4];FOR J :=1 STEP 1 UNTIL N DO DOUBLE(HDM1F[J],HDM1FL[J],FH[I1
248 ,J],FHL[I1,J],T1,T1L,x,*,*,FH[I2,J],FHL[I2,J],T2,T2L,x,*,*,HDM1F[J],HDM1F
249 L[J])END ;FOR J :=1 STEP 1 UNTIL N DO BEGIN HDM1FMID[J]:=HDM1F[J];HDM1FM
250 IDL[J]:=HDM1FL[J]END ;DFLAG :=FALSE ;ACCEPT:FOR I :=1 STEP 1 UNTIL M DO
251 BEGIN CYI :=(INDX :=(INDX +1)MOD TQPI)+TQPI ;DOUBLE(XF,XFL,C2,C2L,I,0,*,
252 H,HL,x,*,*,X,XL);I1 :=(CYI -1)MOD TQPI ;T1 :=PCOEFFL[0];T1L :=PCOEFFL[0]
253 ;T2 :=CCOEFFL[I];T2L :=CCOEFFL[I];FOR J :=1 STEP 1 UNTIL N DO BEGIN T3 :=
254 FH[I1,J];T3L :=FHL[I1,J];DOUBLE(H,HL,T3,T3L,x,HDM1F[J],HDM1FL[J],*,*,T4
255 ,T4L);HDM1F[J]:=T4 ;HDM1FL[J]:=T4L ;DOUBLE(T1,T1L,T3,T3L,x,T4,T4L,*,*,Y
256 P[J],YPL[J]);DOUBLE(T2,T2L,T3,T3L,x,T4,T4L,*,*,CSL[J],CSL[J])END ;I3 :=C
257 YI -QPI ;FOR K :=2 STEP 1 UNTIL M DO BEGIN I1 :=(CYI -K)MOD TQPI ;I2 :=(
258 I3 +K)MOD TQPI ;T1 :=PCOEFFL[I4 :=K -1];T1L :=PCOEFFL[I4];T2 :=CCOEFFL[K];
259 T2L :=CCOEFFL[K];T3 :=PCOEFFL[I4 :=Q -K];T3L :=PCOEFFL[I4];T4 :=CCOEFFL[I4
260 :=QPI -K];T4L :=CCOEFFL[I4];FOR J :=1 STEP 1 UNTIL N DO BEGIN T5 :=FH[I
261 1,J];T5L :=FHL[I1,J];T6 :=FH[I2,J];T6L :=FHL[I2,J];DOUBLE(T1,T1L,T5,T5L,
262 x,T3,T3L,T6,T6L,x,*,*,YPL[J],YPL[J],*,*,YPL[J],YPL[J]);DOUBLE(T2,T2L,T5,T5L
263 ,x,T4,T4L,T6,T6L,x,*,*,CSL[J],CSL[J],*,*,CSL[J],CSL[J])END ;I1 :=(CYI -
264 Q)MOD TQPI ;I2 :=(CYI -QPI)MOD TQPI ;T1 :=PCOEFFL[QMI];T1L :=PCOEFFL[QMI]
265 ;T2 :=CCOEFFL[Q];T2L :=CCOEFFL[Q];T3 :=PCOEFFL[Q];T3L :=PCOEFFL[Q];FOR J :=
266 =1 STEP 1 UNTIL N DO BEGIN T4 :=FH[I1,J];T4L :=FHL[I1,J];DOUBLE(T1,T1L,T
267 4,T4L,x,T3,T3L,FH[I2,J],FHL[I2,J],x,*,*,YPL[J],YPL[J]);DO
268 UBLE(T2,T2L,T4,T4L,x,CSL[J],CSL[J],*,*,CSL[J],CSL[J])END ;T2 :=CCOEFFL[0];

```

```

269 T2L :=CCEFFL[0];CORRECTCNT :=1 ;CORRECT:=F(N,X,XL,YP,YPL,FP,FPL);FOR J :
270 *1 STEP 1 UNTIL N DO BEGIN DOUBLE(T2,T2L,FPL[J],FPL[J],X,CSL[J],CSL[J],+);
271 *T3,T3L:=T3 ;YCL[J]:=T3 ;YCL[J]:=T3L ;DOUBLE(T3,T3L,YPL[J],YPL[J],+);T1,T1L
272 *:=T1 ;:=ABS(T1))>EAV[J]THEN BEGIN IF T1 >ERV[J]*ABS(T3)THEN BEGIN FOR
273 J :=J +1 STEP 1 UNTIL N DO DOUBLE(T2,T2L,FPL[J],FPL[J],X,CSL[J],CSL[J],+);
274 *YCL[J],YCL[J]]:=F(N,X,XL,YP,YPL,FP,FPL);FOR J :=1 STEP 1 UNTIL N DO BEG
275 IN DOUBLE(T2,T2L,FPL[J],FPL[J],X,CSL[J],CSL[J],+);T3,T3L:=T3 ;YPL
276 [J]:=T3L ;DOUBLE(T3,T3L,YCL[J],YCL[J],+);T1,T1L:=T1 ;:=T1 ;:=T1L ;IF T1 >ERV[J]*ABS(T3)T
277 HEN BEGIN IF (T1 :=ABS(T1))>EAV[J]THEN BEGIN FOR J :=J +1 STEP 1 UNTIL N
278 DO DOUBLE(T2,T2L,FPL[J],FPL[J],X,CSL[J],CSL[J],+);YPL[J],YPL[J]]:=CORRECT
279 CNT :=CORRECTCNT +2 ;IF CORRECTCNT >8 THEN BEGIN INDX :=(CYI -1)MOD TQPI
280 ;GO TO L5 END ;GO TO CORRECT END END ;F(N,X,XL,YP,YPL,FH,INDX,*J),FH
281 L(INDX,*J))PFLAG :=TRUE ;CORRECTCNT :=CORRECTCNT +1 ;GO TO L2 END END EN
282 D ;F(N,X,XL,YP,YCL,FH,INDX,*J),FH(INDX,*J))PFLAG :=FALSE ;L2:END ;L1 :=C
283 CYI -N)MOD TQPI ;T1 :=MCEFFL[M];T1L :=MCEFFL[M];FOR J :=1 STEP 1 UNTIL
284 N DO DOUBLE(T1,T1L,FH[L1,J],FH[L1,J],X,HDM1FMIDL[J],HDM1FMIDL[J],+);YM
285 [J],YML[J]]:=CYI -Q ;FOR K :=0 STEP 1 UNTIL M-1 DO BEGIN L1 :=(CYI -
286 K)MOD TQPI ;L2 :=(L1 +K)MOD TQPI ;T1 :=MCEFFL[K];T1L :=MCEFFL[K];T2 :=M
287 CCEFFL[L1 +Q -K];T2L :=MCEFFL[L1 +Q];FOR J :=1 STEP 1 UNTIL N DO DOUBLE(T1
288 *T1L,FH[L1,J],FH[L1,J],X,T2,T2L,FH[L2,J],FH[L2,J],X,+,YML[J],YML[J],+);
289 *YML[J],YML[J])END ;IF DFLAG THEN BEGIN FOR J :=1 STEP 1 UNTIL N DO BEGI
290 N T3 :=Y[J];DOUBLE(T3,YL[J],YML[J],YML[J],+);T2,T2L:=T2 ;:=T2 ;:=T2L ;IF (T2 :=ABS(T2))>
291 EAV[J]THEN BEGIN IF T2 >ERV[J]*ABS(T3)THEN BEGIN IF T2 >EAV[J]THEN BEG
292 IN IF T2 >ERV[J]*ABS(T3)THEN GO TO L4 END ;GO TO L3 END END ;DOUBLE(C
293 C2,C2L,M,0,*,*,*,C2,C2L);DOUBLE(C2,C2L,2,0,*,*,*,C2,C2L);IF PFLAG THEN FO
294 R J :=1 STEP 1 UNTIL N DO BEGIN YL[J]:=YPL[J];YL[J]:=YPL[J]END ELSE FOR J
295 :=1 STEP 1 UNTIL N DO BEGIN YL[J]:=YCL[J];YL[J]:=YCL[J]END ;C1 :=C1 DIV 2
296 ;IF C2 <M THEN GO TO CLOSER ;INDX :=INDX +1 ;FOR K :=1 STEP 1 UNTIL Q DO
297 BEGIN INDX :=(CINDX +1)MOD TQPI ;L1 :=(CINDX +K)MOD TQPI ;FOR J :=1 STEP
298 1 UNTIL N DO BEGIN FH(INDX,J):=FH[L1,J];FH(INDX,J):=FH[L1,J]END END ;G
299 O TO DOUBLER END ;J :=0 ;L3:FOR J :=J +1 STEP 1 UNTIL N DO BEGIN T3 :=YI
300 J];DOUBLE(T3,YL[J],YML[J],YML[J],+);T2,T2L:=T2 ;:=T2 ;:=T2L ;IF (T2 :=ABS(T2))>EAV[J]THE
301 N BEGIN IF T2 >ERV[J]*ABS(T3)THEN BEGIN L4:INDX :=(CYI -M)MOD TQPI ;L5:C
302 1 :=C1 +C1 ;DOUBLE(XF,XFL,C2,C2L,H,HL,X,*,*,X,XL);DOUBLE(C2,C2L,C2,C2L,
303 +*,*,C2,C2L);GO TO STARTER END END ;DOUBLE(C2,C2L,M,0,*,*,*,C2,C2L);I
304 F C2 >M THEN BEGIN IF PFLAG THEN FOR J :=1 STEP 1 UNTIL N DO BEGIN YMID1
305 [J]:=Y[J];YMID1L[J]:=YL[J];YL[J]:=YPL[J];YML[J]:=YML[J];HDM1FMIDL[J]:=HDM1FI
306 J];HDM1FMIDL[J]:=HDM1FMIDL[J]END ELSE FOR J :=1 STEP 1 UNTIL N DO BEGIN YMI
307 D1L[J]:=Y[J];YMID1L[J]:=YL[J];YL[J]:=YCL[J];YML[J]:=YCL[J];HDM1FMIDL[J]:=HDM1
308 FL[J];HDM1FMIDL[J]:=HDM1FMIDL[J]END ;DFLAG :=TRUE ;GO TO ACCEPT END ;IF PFLA

```

[illegible]

```

349 OR I+1STEP 1UNTIL M DO BEGIN L+L+1;ADD2(N,CDL,FVI,YCD);END;FOR K+1STEP 1
350 UNTIL N DO BEGIN DOUBLE(YCK,YPK,TEMPD);IF TEMP#0THEN BEGIN ES+ABS(TE
351 MP)*EFACITOR;IF ES2EACKJTHEN IF ES2ABS(YC(KJ))XERIKJTHEN BEGIN DSW+FALSE;C
352 I+TWOC1;TWOC1+C1+C1;DOUBLE(C2D,TWOD,X+X,C2D);DXD+DXH;DXDL+DXHL;DOUBLE(CXT
353 D,TWOC1D,X+X,DXHD);EFACTOR+C1+P;IF CFSW THEN BEGIN CFSW+FALSE;FOR I+OSTE
354 P 1UNTIL NCF1 DO BEGIN J+I+NCF;DOUBLE(CFI,DXHD,X+X,CJ);END;END ELSE BEGI
355 N CFSW+TRUE;FOR I+OSTEP 1UNTIL NCF1 DO DOUBLE(CFI,DXHD,X+X,CJ);END;DOUBL
356 E(XFD,C2D,HALF,X+X,DXDD,X+X,XMD);FOR K+1STEP 1UNTIL N DO BEGIN YP(KJ)+YMC
357 KJ;YPL(KJ)+YML(KJ);END;GO TO L2;END;IF DSW THEN IF ES2EAKJXERANGE THEN IF
358 ES2ABS(YC(KJ))XERIKJXERANGE THEN DSW+FALSE;END;DOUBLE(C2D,ONE,X+X,C2
359 D);FOR K+1STEP 1UNTIL N DO BEGIN YP(KJ)+YCL(KJ);END;IF C2#0TH
360 EN GO TO EXIT;DOUBLE(XFD,C2D,DXDD,X+X,DXD);IF DSW THEN IF C2>1THEN BEGI
361 N TWOC1+C1+C1+DIV 2;DOUBLE(C2D,TWOD,X+X,C2D);DXH+DXD;DXHL+DXDL;DOUBLE(C
362 DXD,C1D,X+X,DXDD);EFACTOR+C1+P;IF CFSW THEN BEGIN CFSW+FALSE;FOR I+OSTE
363 P 1UNTIL NCF1 DO DOUBLE(CFI,DXDD,X+X,CJ);END ELSE BEGIN CFSW+TRUE;FOR I+
364 OSTEP 1UNTIL NCF1 DO BEGIN J+I+NCF;DOUBLE(CFI,DXDD,X+X,CJ);END;END;END;I
365 F C2<1THEN BEGIN DOUBLE(C2D,ONE,X+X,TEMPD);IF TEMP#0THEN BEGIN EFACTOR+C
366 C1/C2)*P;C1+C2+1;TWOC1+2;DOUBLE(XFD,XD,X+X,DXTD);DXD+DXT;DXDL+DXTL
367 ;DOUBLE(CXDD,TWOD,X+X,DXHD);CFSW+FALSE;FOR I+OSTEP 1UNTIL NCF1 DO BEGIN J
368 +I+NCF;DOUBLE(CFI,DXDD,X+X,CJ);DOUBLE(CFI,DXHD,X+X,CJ);END;END;DOUBL
369 E(XFD,C2D,HALF,X+X,DXDD,X+X,XMD);GO TO L1;EXIT;END;

```

```

01274000
01275000
01276000
01277000
01278000
01279000
01280000
01281000
01282000
01283000
01284000
01285000
01286000
01287000
01288000
01289000
01290000
01291000

FORMAT FMINTOVR; INTEGER OVERFLOW);
LABEL L39;
REAL 800;
INTEGER J;
INTEGER HST;
FILE AB31HST DISK RANDOM "REMOTE" A831+D*(1,90);
MONITOR INTOVR;
FORMAT F211(2I1,1I10);
FORMAT F4E11(4E20,9,19);
DEFINE AC =ALLJ
BEGIN
  DOUBLE(YLJJ,LYJJ,YIJJ,LYIJJ,X+X,TEMP,TEMP);
  TEMP:=ABS(TEMP);
  IF TEMP > 2XEALJJ AND TEMP > ER1JJXABS(YLJJ,YIJJ) THEN WRITE(PUN,
    F4E11,A,C,(YIJJ+YIJJ)/2,TEMP,J);
END #;
FORMAT FM5110 (4(2I1,18,X2),2(I1,18,X2),2(I1,10));

```

```

REAL LA,LC,LDDX;
REAL A,C,DDX;
OWN ARRAY COEF,LCOEF[0:3,0:15,0:89];
INTEGER ARRAY Q[2:73];
OWN INTEGER ARRAY AT,BJ[0:4];
LIST LST(FOR J:=2 STEP 1 UNTIL 72 DO Q[J]);
DEFINE HEREBEFORE =AT[4]#ALLJ =FOR J :=1 STEP 1 UNTIL N DO #
YOBYY =ALLJ
BEGIN
  YO[J]:=Y[J];
  LYO[J]:=LY[J]
END #,YOBYYAV =ALLJ
BEGIN
  DOUBLE(Y[J],LY[J],Y1[J],LY1[J],2,0,0,0,Y[J],LY[J]);
  YO[J]:=Y[J];
  LYO[J]:=LY[J];
END #,SETY1ANDY =ALLJ
BEGIN
  Y1[J]:=Y[J];
  Y[J]:=YO[J];
  LY1[J]:=LY[J];
  LY[J]:=LYO[J]
END #;
LABEL L29,L1,L2,L3,L4,L5,L6,L7,M00,M01,M02,M03,M10,M11,M12,M13,M20,
,M21,M22,M23,M30,M31,M32,M33;
SWITCH SW :=M00,M01,M02,M03,M10,M11,M12,M13,M20,M21,M22,M23,M30,
M31,M32,M33;
SWITCH RETURN :=L1,L2,L3,L4,L5,L6,L7;
INTEGER L,ALF,ALF1,ALF2,BET,BET1,BET2,ALFBEST,BEST0,TALF,TALF1,
TALF2,TBET,TBET1,TBET2,BETBEST,BESTM;
ARRAY LYO,LY1,YO,Y1[0:N];
PROCEDURE RECORDORDERS(M,X,XT,Y,YT,Q);
VALUE M,X,XT,Y,YT;
INTEGER M,X,XT,Y,YT;
ARRAY Q[*];
BEGIN
  INTEGER KX,KY;
  KX :=KY :=1 +8*M;

```

```

KX :=KX +2*X;
KY :=KY +2*Y;
IF XT <YT THEN
BEGIN
  Q[KY+1]:=Q[KY+1]-YT;
  Q[KX J]:=Q[KX J]+XT;
END;
IF YT <XT THEN
BEGIN
  Q[KX+1]:=Q[KX+1]-XT;
  Q[KY J]:=Q[KY J]+YT;
END;
END;

END;

PROCEDURE RECORDMETHODS(X,XT,Y,YT,Q);
VALUE X,XT,Y,YT;
INTEGER X,XT,Y,YT;
ARRAY Q[*];
BEGIN
  INTEGER KX33,KX41,KY33,KY41;
  KX33 :=33 +2*X;
  KY33 :=33 +2*Y;
  KX41 :=8 +KX33 +8*X;
  KY41 :=8 +KY33 +8*Y;
  IF XT <YT THEN
  BEGIN
    Q[KY33+1]:=Q[KY33+1]-YT;
    Q[KX33 J]:=Q[KX33 J]+XT;
    Q[KY41+1]:=Q[KY41+1]-YT;
    Q[KX41 J]:=Q[KX41 J]+XT;
  END;
  IF YT <XT THEN
  BEGIN
    Q[KX33+1]:=Q[KX33+1]-XT;
    Q[KY33 J]:=Q[KY33 J]+YT;
    Q[KX41+1]:=Q[KX41+1]-XT;
  END;

```

```

Q[KY41 J]:=Q[KY41 J+YT];
END;

END;
PROCEDURE SECTORDERS(M,X,Y,Q);
INTEGER M,X,Y;
ARRAY Q[*];
BEGIN
  INTEGER J,S,K;
  J:=Q[0]:(Q[0]*4093 +3000001)MOD 16777216 ;
  J :=J/@4;
  FOR S :=0 STEP 1 UNTIL 3 DO
    BEGIN
      J :=(J+1)MOD 4;
      K :=1 +8XM +2xJ;
      AT[S]:=Q[K]+Q[K+1];
      BJ[S]:=J;
    END;
  X:=BJ[0];
  Y :=IF AT[1]>AT[2]THEN IF AT[1]>AT[3]THEN BJ[1]ELSE BJ[3]ELSE IF
    AT[2]>AT[3]THEN BJ[2]ELSE BJ[3];
  IF AT[0]<AT[1]THEN IF AT[0]<AT[2]THEN IF AT[0]<AT[3]THEN
    SECTORDERS(M,X,Y,Q);
  END;
END;
PROCEDURE SECTMETHODDS(X,Y,Q);
INTEGER X,Y;
ARRAY Q[*];
BEGIN
  INTEGER J,S,K;
  J:=Q[0]:(Q[0]*4093 +3000001)MOD 16777216 ;
  J :=J/@4;
  FOR S :=0 STEP 1 UNTIL 3 DO
    BEGIN
      J :=(J+1)MOD 4;
      K :=33 +2xJ;

```

```

      AT[S]:=Q[K]+Q[K+1];
      BJ[S]:=J;
    END;
    X:=BJ[0];
    Y:=IF AT[1]≥AT[2] THEN IF AT[1]≥AT[3] THEN BJ[1] ELSE BJ[3] ELSE IF 01412000
    AT[2]≥AT[3] THEN BJ[2] ELSE BJ[3]; 01413000
    IF AT[0]≤AT[1] THEN IF AT[0]≤AT[2] THEN IF AT[0]≤AT[3] THEN 01414000
    SELECTMETHODS(X,Y,Q); 01415000
  END;
  X:=BJ[0];
  Y:=IF AT[1]≥AT[2] THEN IF AT[1]≥AT[3] THEN BJ[1] ELSE BJ[3] ELSE IF 01416000
  AT[2]≥AT[3] THEN BJ[2] ELSE BJ[3]; 01417000
  IF AT[0]≤AT[1] THEN IF AT[0]≤AT[2] THEN IF AT[0]≤AT[3] THEN 01418000
  SELECTMETHODS(X,Y,Q); 01419000
END; 01420000
IF HEREBEFORE # "YES" THEN 01421000
BEGIN 01422000
  FORMAT FM ("SEND THIS HISTORY FILE ",WTO L J GALLAHER VIA ",WCAM 01423000
  PUS MAIL"//X5,A5,I10,I10/(8T10)); 01424000
  FILE IN TAPE831 DISK SERIAL "REMOTE" TAPE831"(2,90); 01425000
  INTEGER J,K,S,LIM; 01426000
  COMMENT READ COEF.FILE; 01427000
  FOR J:=0 STEP 1 UNTIL 3 DO 01428000
  BEGIN 01429000
    LIM:=IF J=0 THEN 15 ELSE IF J=1 THEN 5 ELSE 6; 01430000
    FOR K:=0 STEP 1 UNTIL LIM DO 01431000
    BEGIN 01432000
      READ(TAPE831,89,COEF[J,K,*]); 01433000
      READ(TAPE831,89,LCOEF[J,K,*]); 01434000
    END; 01435000
  END; 01436000
  HEREBEFORE := "YES"; 01437000
  WRITE(PUN,FM); 01438000
END; 01439000
BEGIN 01440000
  REAL A,C; 01441000
  INTEGER J,S,K; 01442000
  A:=850; 01443000
  FOR J:=1 STEP 1 UNTIL N DO IF EA[J]≤A AND ER[J]≤A THEN A:=(EA 01444000
  [J]+ER[J])/2; 01445000
END; 01446000
BEGIN 01447000
  REAL A,C; 01448000
  INTEGER J,S,K; 01449000
  A:=850; 01450000
  FOR J:=1 STEP 1 UNTIL N DO IF EA[J]≤A AND ER[J]≤A THEN A:=(EA 01451000
  [J]+ER[J])/2;

```

```

J :=IF A>9.999@=9 THEN 0 ELSE IF A>@=14 THEN 1 ELSE 2;
C :=TIME(2);
F(N,XI,LXI,Y,YO,LYO);
C :=(TIME(2)-C)/N;
K :=IF C<1.01 THEN 0 ELSE 1;
S :=IF N<7 THEN 0 ELSE 1;
HST :=K +2X(S +2XJ);
READ(A831HST[HST],75,Q[*]);
B00 :=3@2 x10*J;
SELECTMETHODS(ALF,BET,Q);
SELECTORDERS(ALF,ALF1,ALF2,Q);
SELECTORDERS(BET,BET1,BET2,Q);

END;
INTOVR :=L39;
A :=XI;
LA :=LXI;
DOUBLE(XF,LXF,XI,LXI,=8.0,/=DDX,LDDX);
Y0BY;
DOUBLE(A,LA,DDX,LDDX,+=C,LC);
WRITE(PUN,F2I1,ALF,ALF1,HST);
TALF1 :=TIME(2);
L:=1;
GO TO SW[4xALF+ALF1+1];
L1:TALF1 :=TIME(2)-TALF1;
SETY1ANDY;
WRITE(PUN,F2I1,ALF,ALF2);
TALF2 :=TIME(2);
L:=2;
GO TO SW[4xALF+ALF2+1];
L2:TALF2 :=TIME(2)-TALF2;
AC;
Y0BYAV;
A :=C;
LA :=LC;
DOUBLE(A,LA,DDX,LDDX,+=C,LC);
WRITE(PUN,F2I1,BET,BET1);
TBET1 :=TIME(2);
L:=3;
GO TO SW[4xBET+BET1+1];

```

```

01452000
01453000
01454000
01455000
01456000
01457000
01458000
01459000
01460000
01461000
01462000
01463000
01464000
01465000
01466000
01467000
01468000
01469000
01470000
01471000
01472000
01473000
01474000
01475000
01476000
01477000
01478000
01479000
01480000
01481000
01482000
01483000
01484000
01485000
01486000
01487000
01488000
01489000
01490000
01491000

```

```

L3:TBET1 :=TIME(2)-TBET1;
SETY1ANDY;
WRITE(PUN ,F2I1,BET,BET2);
TBET2 :=TIME(2);
L:=4;
GO TO SW[4xBET+BET2+1];
L4:TBET2 :=TIME(2)-TBET2;
AC;
YOBYYAV;
A :=C;
LA :=LC;
DOUBLE(2,0,DDX,LDDX,x,A,LA,+,+,C,LC);
ALFBEST :=IF TALF1<TALF2 THEN ALF1 ELSE ALF2;
BETBEST :=IF TBET1<TBET2 THEN BET1 ELSE BET2;
WRITE(PUN ,F2I1,ALF,ALFBEST);
TALF :=TIME(2);
L:=5;
GO TO SW[4*ALF+ALFBEST+1];
L5:TALF :=TIME(2)-TALF ;
SETY1ANDY;
WRITE(PUN ,F2I1,BET,BETBEST);
TBET :=TIME(2);
L:=6;
GO TO SW[4xBET+BETBEST+1];
L6:TBET :=TIME(2)-TBET ;
AC;
YOBYYAV;
A :=C;
LA :=LC;
C :=XF;
LC :=LXF;
FOR J :=1 STEP 1 UNTIL 40 DO Q[J]:=Q[J]*0.980;
RECORDORDERS(ALF,ALF1,TALF1,ALF2,TALF2,Q);
RECORDORDERS(BET,BET1,TBET1,BET2,TBET2,Q);
RECORDMETHODS(ALF,TALF,BET,TBET,Q);
Q[-2]:=TIME(0);
Q[-1]:=Q[-1]+1;
WRITE(A831HST[HST],75,Q[*]);
WRITE(PUN ,FM5I10,ALF,ALF1,TALF1,ALF,ALF2,TALF2,BET,BET1,TBET1,BET1530000
,BET2,TBET2,ALF,TALF,BET,TBET,Q[-1],HST);

```

```

IF TALF ≤TBET THEN
BEGIN
  BESTM :=ALF;
  BESTO :=ALFBEST
END ELSE
BEGIN
  BESTM :=BET;
  BESTO :=BETBEST
END;
L :=7;
GO TO SW[4×BESTM +BESTO +1];

BEGIN
GO TO L29;
M00:COMMENT;
DADAMS(N,A,LA,C,LC,Y,LY,F,P,9,DX,EA,ER,COEF[0,6,*],LCDEF[0,6,*])
,9,7,COEF[3,4,*],LCDEF[3,4,*],DSTART,DSHANKS);
GO TO L29;
M01:COMMENT;
DADAMS(N,A,LA,C,LC,Y,LY,F,P,10,DX,EA,ER,COEF[0,7,*],LCDEF[0,7,*],
,9,7,COEF[3,4,*],LCDEF[3,4,*],DSTART,DSHANKS);
GO TO L29;
M02:COMMENT;
DADAMS(N,A,LA,C,LC,Y,LY,F,P,12,DX,EA,ER,COEF[0,9,*],LCDEF[0,9,*],
,9,7,COEF[3,4,*],LCDEF[3,4,*],DSTART,DSHANKS);
GO TO L29;
M03:COMMENT;
DADAMS(N,A,LA,C,LC,Y,LY,F,P,13,DX,EA,ER,COEF[0,10,*],LCDEF[0,10,*],
,J,9,7,COEF[3,4,*],LCDEF[3,4,*],DSTART,DSHANKS);
GO TO L29;
M10:COMMENT;
BUTCHER(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,9,7,COEF[3,4,*],LCDEF[3,4,*],
,J,B00,3,COEF[1,2,*],LCDEF[1,2,*],DSTART,DSHANKS);
GO TO L29;
M11:COMMENT;
BUTCHER(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,9,7,COEF[3,4,*],LCDEF[3,4,*],
,J,B00,4,COEF[1,3,*],LCDEF[1,3,*],DSTART,DSHANKS);
GO TO L29;
M12:COMMENT;
BUTCHER(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,9,7,COEF[3,4,*],LCDEF[3,4,*],

```

01532000

01533000

01534000

01535000

01536000

01537000

01538000

01539000

01540000

01541000

01542000

01543000

01544000

01545000

01546000

01547000

01548000

01549000

01550000

01551000

01552000

01553000

01554000

01555000

01556000

01557000

01558000

01559000

01560000

01561000

01562000

01563000

01564000

01565000

01566000

01567000

01568000

01569000

01570000

01571000

```

*J,B00,5,COEF[1,4,*],LCOEF[1,4,*],DSTART,DSHANKS)),
GO TO L29,
01572000
M13:COMMENT,
01573000
BUTCHER(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,12,8,COEF[3,6,*],LCOEF[3,6,*],
01574000
,*,J,B00,6,COEF[1,5,*],LCOEF[1,5,*],DSTART,DSHANKS)),
01575000
GO TO L29,
01576000
M20:COMMENT,
01577000
COWELL(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,5,5,COEF[3,1,*],LCOEF[3,1,*],
01578000
,*,J,B00,8,COEF[1,5,*],LCOEF[1,5,*],DSTART,DSHANKS)),
01579000
GO TO L29,
01580000
M21:COMMENT,
01581000
COWELL(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,9,7,COEF[3,4,*],LCOEF[3,4,*],
01582000
,*,J,B00,10,COEF[1,5,*],LCOEF[1,5,*],DSTART,DSHANKS)),
01583000
GO TO L29,
01584000
M22:COMMENT,
01585000
COWELL(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,9,7,COEF[3,4,*],LCOEF[3,4,*],
01586000
,*,J,B00,12,COEF[1,5,*],LCOEF[1,5,*],DSTART,DSHANKS)),
01587000
GO TO L29,
01588000
M23:COMMENT,
01589000
COWELL(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,9,7,COEF[3,4,*],LCOEF[3,4,*],
01590000
,*,J,B00,14,COEF[1,5,*],LCOEF[1,5,*],DSTART,DSHANKS)),
01591000
GO TO L29,
01592000
M30:COMMENT,
01593000
DSHANKS(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,7,7,COEF[3,3,*],LCOEF[3,3,*],P,EA,
01594000
,ER,DX)),
01595000
GO TO L29,
01596000
M31:COMMENT,
01597000
DSHANKS(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,9,7,COEF[3,4,*],LCOEF[3,4,*],P,EA,
01598000
,ER,DX)),
01599000
GO TO L29,
01600000
M32:COMMENT,
01601000
DSHANKS(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,10,8,COEF[3,5,*],LCOEF[3,5,*],P,
01602000
EA,ER,DX)),
01603000
GO TO L29,
01604000
M33:COMMENT,
01605000
DSHANKS(N,A,LA,C,LC,Y,LY,F,EA,ER,P,DX,12,8,COEF[3,6,*],LCOEF[3,6,*],P,
01606000
EA,ER,DX)),
01607000
GO TO L29,
01608000
L39:WRITE(PUN,FMINTOVR),
01609000
ALLJ
01610000
01611000

```

```

01612000
01613000
01614000
01615000
01616000
01617000
01618000
01619000
01620000
01621000

```

```

BEGIN
  LY[J]:=LYO[J];
  Y [J]:=YO[J];
END;
DIFEQINT(N,A,LA,C,LC,Y,LY,F,P,EA,ER,DX);
L29:GO TO RETURN[L];
END;
L7:
END;

```

```

COMMENT "
00000000
00001000
00002000
00003000
00004000
";
00005000
00006000
00007000
00008000
00009000
00010000
00011000
00012000
00013000

FILE PUN 15 "HISTORY"(3,15);
PROCEDURE DIFEQINT(N,XI,XF,Y,F,P,E,A,E,R,D,X);
VALUE N,XF,XI,DX,P;
INTEGER N;
REAL XI,XF,DX,P;
PROCEDURE F;
ARRAY Y,E,A,E,R(0);
IF XI #XF THEN
BEGIN
INTEGER PROCEDURE START(N,XI,XF,C1,E,A,E,R,F,M,0000 8
,P,A,P,FNEVAL;INTEGER N,C1,M,CYI,CYM,PA,P,FNEVAL,ARKSCONST);VALUE N,XI,XF,C1,M,CYI,CYM0000 9
AY E,A,E,R,YIV,YFV,ARKSCONST(0),YH,FH(0,0);PROCEDURE F;BEGIN INTEGER I,J,K0000 10
,L,COEFFCNT,FNMAX,INDEX,NINDX,CNTR;REAL INT,M,TWOH,T1;REAL ARRAY HC,TWO0000 11
HC(0:((FNEVAL+3)*FNEVAL)/2)=2;EAV,ERV,YI,Y2,Y3,Y4(0:N),G(0:FNEVAL-2,0000 12
,N);LABEL L1,L2,L3,L4,L5,L6;PROCEDURE RUNKUT(N,X,FNMAX,COEFF,YIV,YFV,FV0000 13
,F,G);VALUE N,X,FNMAX;INTEGER N,FNMAX;REAL X;REAL ARRAY COEFF,YIV,YFV0000 14
,FV(0),G(0,0);PROCEDURE F;BEGIN INTEGER I,J,K,CNTR;REAL TEMP;TEMP :=C0000 15
DEFFCNT :=0;FOR I :=0 STEP 1 UNTIL FNMAX DO BEGIN FOR J :=1 STEP 1 UN0000 16
TIL N DO YFV(J):=FV(J)*TEMP +YIV(J);FOR K :=0 STEP 1 UNTIL I-1 DO BEGIN0000 17
TEMP :=COEFFCNT :=CCNT +1;FOR J :=1 STEP 1 UNTIL N DO YFV(J):=G(K,J)0000 18
XTEMP +YFV(J)END;FCN,X +COEFFCNT :=CCNT +1,YFV,G(I,*));TEMP :=COEFFC0000 19
CNT :=CCNT +1;END;FOR J :=1 STEP 1 UNTIL N DO YFV(J):=FV(J)*TEMP +YIV(0000 20
J);FOR K :=0 STEP 1 UNTIL FNMAX DO BEGIN TEMP :=COEFFCNT :=CCNT +1;F0000 21
R J :=1 STEP 1 UNTIL N DO YFV(J):=G(K,J)*TEMP +YFV(J)END;END;BOOLEAN P0000 22
RUCEDURE COMP(N,EAV,ERV,Y,Z);VALUE N;INTEGER N;REAL ARRAY EAV,ERV,Y,Z(0000 23
0);BEGIN INTEGER J;REAL T1;LABEL L1;FOR J :=1 STEP 1 UNTIL N DO IF (T0000 24
1 :=ABS(Y(J)-Z(J))>EAV(J)THEN BEGIN IF T1 >ERV(J)*ABS(Z(J))THEN BEGIN C0000 25
OMP :=FALSE;GO TO L1;END;END;COMP :=TRUE;L1:END;CNTR :=1;IF M #0 TH0000 26
EN BEGIN COEFFCNT :=((FNEVAL +3)*FNEVAL)/2)-2;FNMAX :=FNEVAL-2;INT :=0000 27
XF "XI;TWOH :=(INT +INT)/C1;H :=INT /C1;FOR I :=0 STEP 1 UNTIL COEFF0000 28
CNT DO BEGIN HC(I):=(T1 :=RKSCONST(I))*H;TWOHCL(I):=T1 *TWOH;END;INDX :=0000 29
:=CYI MOD CYM;T1 :=(C1 /2)*P;FOR J :=1 STEP 1 UNTIL N DO BEGIN EAV(J):=0000 30
31

```

```

EA[J]/T1 ;ERV[J]:=ER[J]/T1 END ;IF PA =2 THEN BEGIN NINDX :=(INDX +1)MOD0000 32
CYM ;K :=0 ;GO TO L3 END ;IF PA =1 THEN L :=M DIV 2 ;RUNKUT(N,X,FNMAX,T0000 33
WUHC,YIV,Y1,FH[INDX],F,G);GO TO L4 ;L1:TWOH :=H ;CNTR :=CNTR +CNTR ;C10000 34
:=C1 +C1 ;H :=INT /C1 ;FOR I :=0 STEP 1 UNTIL COEFFCNT DO BEGIN TWOHC[I]0000 35
J:=HC[I];HC[I]:=RKSCONST[I]*H END ;INDX :=CYI MOD CYM ;T1 :=(C1 /2)*P ;I0000 36
F PA =2 THEN BEGIN K :=0 ;NINDX :=(INDX +1)MOD CYM ;FOR J :=1 STEP 1 UNT0000 37
IL N DO BEGIN EAV[J]:=EAV[J]/T1 ;ERV[J]:=ER[J]/T1 ;YI[J]:=YH[NINDX,J]END 0000 38
;L2:RUNKUT(N,K,X,X,FNMAX,HC,YH[INDX],F,G);YH[NINDX,J]:=F,G);K :=0000 39
:=K +1 ;INDX :=NINDX ;NINDX :=(INDX +1)MOD CYM ;F(N,K,X,X,YH[INDX],F0000 40
[INDX,J];RUNKUT(N,K,X,X,FNMAX,HC,YH[INDX],F,G);YH[NINDX,J]:=F,G);F,0000 41
G);IF COMP(N,EAV,ERV,Y1,YH[NINDX,J])THEN BEGIN K :=K +1 ;IF K <M THEN BE0000 42
GIN INDX :=NINDX ;NINDX :=(INDX +1)MOD CYM ;F(N,K,X,X,YH[INDX],F,0000 43
DX,J);L3:RUNKUT(N,K,X,X,FNMAX,TWOHC,YH[INDX],F,G);YI[FH[INDX],F,G];GO 0000 44
TO L2 END ;IF K =M THEN BEGIN INDX :=NINDX ;F(N,K,X,X,YH[INDX],F,0000 45
DX,J)END ELSE GO TO L1 END ELSE BEGIN FOR J :=1 STEP 1 UNTIL N DO B0000 46
EGIN EAV[J]:=EAV[J]/T1 ;ERV[J]:=ER[J]/T1 ;YI[J]:=Y2[J]END ;L4:RUNKUT(N,X,0000 47
FNMAX,HC,YIV,Y2,FH[INDX],F,G);INDX :=(INDX +1)MOD CYM ;F(N,X,X,Y2,FH[0000 48
INDX,J];RUNKUT(N,X,X,H,FNMAX,HC,Y2,Y4,FH[INDX],F,G);K :=2 ;IF PA =0 TH0000 49
EN BEGIN L5:IF COMP(N,EAV,ERV,Y1,Y4)THEN BEGIN IF K <M THEN BEGIN INDX :=0000 50
=(INDX +1)MOD CYM ;F(N,K,X,X,Y4,FH[INDX],F,G);RUNKUT(N,K,X,X,FNMAX,TW0000 51
HC,Y4,Y1,FH[INDX],F,G);RUNKUT(N,K,X,X,FNMAX,HC,Y4,Y3,FH[INDX],F,G)0000 52
;K :=K +1 ;INDX :=(INDX +1)MOD CYM ;F(N,K,X,X,Y3,FH[INDX],F,G);RUNKUT(N,0000 53
K,X,X,FNMAX,HC,Y3,Y4,FH[INDX],F,G);K :=K +1 ;GO TO L5 END ;IF K =M T0000 54
HEN BEGIN INDX :=(INDX +1)MOD CYM ;F(N,K,X,X,Y4,FH[INDX],F,G);FOR J :=1 0000 55
STEP 1 UNTIL N DO YFV[J]:=Y4[J]END ELSE FOR J :=1 STEP 1 UNTIL N DO YFV[0000 56
J]:=Y3[J]END ELSE GO TO L1 END ELSE BEGIN L6:IF COMP(N,EAV,ERV,Y1,Y4)THE0000 57
N BEGIN INDX :=(INDX +1)MOD CYM ;F(N,K,X,X,Y4,FH[INDX],F,G);IF K <M TH0000 58
BEGIN IF K =L THEN FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=Y4[J];RUNKUT(N,K0000 59
X,X,FNMAX,TWOHC,Y4,Y1,FH[INDX],F,G);RUNKUT(N,K,X,X,FNMAX,HC,Y4,Y30000 60
,FH[INDX],F,G);K :=K +1 ;INDX :=(INDX +1)MOD CYM ;F(N,K,X,X,Y3,FH[IN0000 61
DX,J];IF K =L THEN FOR J :=1 STEP 1 UNTIL N DO YFV[J]:=Y3[J];RUNKUT(N,K0000 62
X,X,FNMAX,HC,Y3,Y4,FH[INDX],F,G);K :=K +1 ;GO TO L6 END ;FOR J :=1 0000 63
STEP 1 UNTIL N DO YIV[J]:=Y4[J]END ELSE GO TO L1 END END ;X :=M *X +X EN0000 64
D ;START :=CNTR ;END ;PROCEDURE ADAMS(N,XI,XF,Y,P,Q,DX,EAV,ER,ADAMSCDEF0000 65
F,RKSFNS,RKSORDE,RKSCOEFF,START,SHANKS);VALUE N,XI,XF,P,Q,DX,RKSFNS,RKS0000 66
ORDER,REAL XI,XF,P,Q,INTEGER N,Q,RKSFNS,RKSORDE,REAL ARRAY Y,EAV,ER,ADA0000 67
MSCOEFF,RKSCOEFF[0];PROCEDURE F,SHANKS;INTEGER PROCEDURE START;BEGIN DEF0000 68
INE YI:=Y#;YB:=Y#;YF:=Y#;POINTS:=BEGIN IF NOT BGOOD THEN ALLI YB[I];X+0000 69
XF=INTERVAL*(C2/C1);MULT+START(N,XI,XF,C1,EAV,F,Q,MINUS1,X,YB,FH,YB,0000 70
J,Q,TM1,0,P,RKSFNS,RKSCOEFF);ALLI C[I];YB[I];YB[I];BGOOD+TRUE;END#;CALL0000 71

```

```

72 B=BEGIN SHANKS(N,X,XF,YB,F,RKSFNS,RKORDER,RKSCOEFF,P,EA,ER,XF=X);END#;A0000
73 LLMU=FOR MU<0STEP 1UNTIL QMINUS1 DO#;ALLI=FOR I<1STEP 1UNTIL N DO#;RESET0000
74 =BEGIN DC<0;PC<0;CU<(C1*(P))XGR;H<INTERVAL/C1;ALLMU BEGIN HB[MU]B[CU]X0000
75 H;HBS[MU]B[CU]X0000;HBSQZ<B[SQZ]H;ALLI BEGIN EAL[I]EAL[I]XCU)X0000
76 C2MQP5;ERL[I]EAL[I]XCU)X0000;HBSQZ<B[SQZ]H;ALLI BEGIN EAL[I]EAL[I]XCU)X0000
77 BS(ERL[I]HBSQZ)END;END#;LABEL RESTART,NEXTSTEP,FLIP,FLOP,TEST,DUBBLE,H0000
78 ALF,FINISH;REAL ARRAY FH[0:(C2XQ-2)];0;N;B;BS;HB;HBS[0:Q-1];C;YP;YC;YO;FP0000
79 FC;EAU;EAL;ERU;ERL;HAL;HRL[0:N];REAL H;X;CU;C2;GR;YCI;BSQZ;FHJI;FMUI;HB0000
80 MU;M1DP;HBSMU;HBSQZ;ERROR;CHANGE;C2MQP5;INTERVAL;INTEGER I;J;K;C1;DC;PC;0000
81 MU;MULT;JZERD;QT2M1;QMINUS1;QTIMES2;BOOLEAN BGOOD;FLIPPED;TOOSMALL;C2MQP0000
82 5<1/2*(Q+5);QMINUS1<Q-1;QTIMES2<Q+Q;QT2M1<QTIMES2-1;ALLMU BEGIN B[CU]AD0000
83 AMSCOEFF[CU]B[S[MU]]ADAMSCOEFFIMU+Q1;END;BSQZ<ADAMSCOEFF[QTIMES2];GR<ADA0000
84 MSCOEFF[QTIMES2+1];C1+1;H<INTERVAL<XF=X;DX<ABS(DX);WHILE ABS(H)>DX OR C0000
85 1<Q DO BEGIN C1+C1<C1;H<INTERVAL/C1;END;C2+C1;JZERD<J<0;F(N,XI,YI,FHCO)*0000
86 J);X<XI;ALLI YB[I]YB[I]XV[I]BGOOD<TRUE;RESTART;POINTS;C1<C1XMULT;C2<C2XMULT0000
87 =Q;IF(J+JZERD=1)<0THEN J+J+QT2M1;RESET;NEXTSTEP;X<XF<C2XH;ALLMU BEGIN IF0000
88 (J+J+1)=QT2M1 THEN J<0;HBMU+HB[MU];HBSMU+HBS[MU];ALLI BEGIN YP[I]CFMUI+0000
89 FHEJ,I);XHBMU+YPII;CII;HBSMU+FMUI+CI;END;END;F(N,X,YP,FP);FLIP;ALLI 0000
90 YCII<FPII;HBSQZ+CI;F(N,X,YC,FC);ALLI IF(CHANGE<ABS(FCII-FPII))>HAL0000
91 I;THEN IF CHANGE>ABS(HRL[I]XFCII);THEN GO TO FLOP;FLIPPED<TRUE;GO TO TE0000
92 ST;FLOP;ALLI YCII<FCII;HBSQZ+CI;F(N,X,YC,FP);ALLI IF(CHANGE<ABS(FCII-FPII))
93 J<FCII);HAL[I];THEN IF CHANGE>ABS(HRL[I]XFCII);THEN GO TO FLIP;FLIPPED<0000
94 FALSE;TEST;IF(J+J+1)=QT2M1 THEN J<0;TOOSMALL<TRUE;ALLI BEGIN FHJI<FHEJ,I)0000
95 J<IF FLIPPED THEN FCI;ELSE FPII;ERROR<ABS(YPII-CYCI);YPII<YPII<CFHJI)X0000
96 HBSQZ+CI);IF BGOOD THEN YCII<YCI ELSE YBII<YCI;YCI<ABS(YCI);IF ERR0000
97 OR>EAU;I;THEN IF ERROR>ERUII;YCI THEN GO TO HALF;IF ERROR>EALCI;THEN IF0000
98 ERROR>ERLII;YCI THEN TOOSMALL<FALSE;END;PC+1;IF BGOOD THEN BGOOD<FA0000
99 LSE ELSE BGOOD<TRUE;IF C2<1,0THEN C2<C2-1;0ELSE GO TO FINISH;IF TOOSMALL0000
100 THEN BEGIN DC<DC+1;IF DC<3;THEN IF PC<QT2M1 THEN IF C2<1;THEN GO TO DUBBL0000
101 E;IF(JZERD<(J+JZERD)+1)=QT2M1 THEN JZERD<0;GO TO NEXTSTEP;END;DC<0;IFCJ20000
102 EK0<(J+JZERD)+1)=QT2M1 THEN JZERD<0;GO TO NEXTSTEP;DUBBLE<C1<C1 DIV 2;C20000
103 <(C2-1,0)/2,0;RESET;K<0;FOR MU<1STEP 1UNTIL QMINUS1 DO BEGIN IF(CJ+J-1)<00000
104 THEN J+J+QT2M1;IF(K<K-2)<0THEN K+K+QT2M1;ALLI FHEJ,I);FHEJ,I);END;IF(CJ+0000
105 JZERD+J)=1)<0THEN J+J+QT2M1;GO TO NEXTSTEP;HALF<FCJ+J-1)<0THEN J+J+QT2M0000
106 1;JZERD<J+J+1;IF(C2<C2+2,0)<0 THEN GO TO FINISH;GO TO RESTART;FI0000
107 NISH;IF NOT FLIPPED THEN ALLI FCI;FPII;IF NOT BGOOD THEN ALLI YBII<Y0000
108 DCII<X<XF<INTERVAL<(C2/C1);IF C2<0THEN CALLOB;ALLI YFIJ<YBII;END;PROC0000
109 DURE BUTCHER(N,XI,XF,K,EAU,ER,DX,CON,FUNCTION,EX,KC,START,SHANKS,YIV,RKS0000
110 NF,RKSODR);VALUE N,XI,XF,K,DX,CON,EX,RKSNF,RKSODR;REAL ARRAY YIV[0];INTE0000
111 GER RKSNF,RKSODR;INTEGER N,K;PROCEDURE FUNCTION,SHANKS;INTEGER PROCEDURE0000

```

```

START;REAL XI,XF,DX,EX;REAL ARRAY RKC(0);REAL ARRAY CON,EA,ER(0);BEGIN 0000 112
REAL ARRAY Y,F(0:16,0:N);REAL SC1,X;REAL DX2;REAL DX1,COA,C08,COLA,COLB,0000 113
CUGA,C0GB,TEST,TEMPY,TEMPF,A1,A2,A3,C2;INTEGER I,J,CYL,INDEX,C1,M;INTEGE0000 114
R CYL3;REAL ARRAY SUMYIP,SUMYP,SUMYC,FV1(0:N);LABEL STRT,RESTART,FINISH0000 115
;REAL P2,T1,T2;INTEGER COUNT,TOTCNT,CYL1,CYL2,M1;LABEL DUBSRT;REAL ARRAY0000 116
C00(0:3XKJ);INTEGER CY0;INTEGER COUNTER;INTEGER KM1;REAL OMT,K6,K61,K62;0000 117
REAL INTV;INTEGER KM3,J2,J3,J6;REAL XDXT,XDX;REAL ARRAY RE,AE(0:N);FOR 10000 118
+1STEP 1UNTIL N DO Y(0,IJ+YIVL(IJ));IF K=10R K=20R K=3THEN OMT+0.5ELSE OMT+0000 119
2/3;K6+6XK;K61+(6XK))+1;K62+(6XK))+2;KM1+K=1;INTV+XF=XI;XC1+1;WHILE(C10000 120
<K+1)OR((ABS(INTV)/C1)>ABS(DX))DO C1+C1+C1;C2+C1;P2+1/(2*((2XK))+4));CYL+0000 121
0;CY0+0;TOTCNT+0;FUNCTION(N,XI,Y(0,*),F(0,*));RESTART;COUNTER+KM1;C1+C1X0000 122
(I+START(N,XI,XF,C1,EA,ER,FUNCTION,KM1,X,YIV,Y,F,YIV,CY0,16,2*EX,RKSNF,R0000 123
KC));C2+C2XI-KM1;CYL+CY0;DUBSRT;DX+INTV/C1;KM3+3XKM1;FOR J+0STEP 3UNTIL 0000 124
KM3 DO BEGIN J2+2XJ;C00(J)+CON(J2+1J+DX);C00(J+1J)+CON(J2+3J+DX);C00(J+2J)+C0000 125
ON(J2+5J+DX);END;SC1+C1+EX;FOR I+1STEP 1UNTIL N DO BEGIN AE(IJ)+EAL(IJ/SC1)0000 126
RE(IJ)+ER(IJ/SC1);END;A1+CON(K6J+DX);A2+CON(K61J+DX);A3+CON(K62J+DX);STRT;X0000 127
XT+X+(DXXOMT);XDX+X+DX;FOR I+1STEP 1UNTIL N DO SUMYIP(IJ)+SUMYP(IJ)+SUMYC0000 128
IJ+0;FOR J+0STEP 1UNTIL KM1 DO BEGIN CYL3+(KM1-J+CYL)MOD 16;J3+3XJ;J6+6X0000 129
J;COA+CON(J6J);CUB+C00(J3J);COLA+CON(J6+2J);COLB+C00(J3+1J);COGA+C0N(J6+4J);C0000 130
OGB+C00(J3+2J);FOR I+1STEP 1UNTIL N DO BEGIN TEMPY+Y(CYL3,IJ);TEMPF+FLCYL30000 131
;IJ;SUMYIP(IJ)+SUMYIP(IJ)+(COAXTEMPY)+(COBXTEMPF);SUMYP(IJ)+SUMYP(IJ)+(COLA0000 132
TEMPY)+(COLBXTEMPF);SUMYC(IJ)+SUMYC(IJ)+(COGAXTEMPY)+(COGBXTEMPF);END;END0000 133
FUNCTION(N,XDXT,SUMYIP,FV1);FOR I+1STEP 1UNTIL N DO BEGIN TEMPF+FV1(IJ);S0000 134
UMYP(IJ)+SUMYP(IJ)+(A1XTEMPF);SUMYC(IJ)+SUMYC(IJ)+(A2XTEMPF);END;FUNCTION(N,0000 135
XDX,SUMYP,FV1);CYL+(CYL+1)MOD 16;CY0+(CYL+KM1)MOD 16;COUNT+0;FOR I+1STEP0000 136
1UNTIL N DO BEGIN TEMPY+SUMYC(IJ)+(A3XFV1(IJ));T1+AE(IJ);T2+ABS(RE(IJ)XTEMP0000 137
Y);TEST+ABS(TEMPY-SUMYP(IJ));IF TEST>T1 AND TEST>T2 THEN BEGIN C2+C2;C0000 138
YL+(CYL+15)MOD 16;CY0+(CYL+KM1)MOD 16;IF C2<KM1 THEN BEGIN SHANKS(N,X,XF0000 139
,Y(CY0,*),FUNCTION,RKSNF,RKSODR,RKC,EX,EA,ER,DX);GO TO FINISH;END;C1+C1+0000 140
C1;GO TO RESTART;END;Y(CY0,IJ)+TEMPY;IF TEST<P2XT1 OR TEST<P2XT2 THEN COU0000 141
NT+COUNT+1;END;C2+C2-1;X+XF=(DXXC2);IF C2=0THEN GO TO FINISH;IF C2<1THEN0000 142
BEGIN SHANKS(N,X,XF,Y(CY0,*),FUNCTION,RKSNF,RKSODR,RKC,EX,EA,ER,DX);GO 0000 143
TO FINISH;END;FUNCTION(N,X,Y(CY0,*),FLCY0,*);IF COUNT=N THEN BEGIN TOTC0000 144
NT+TOTCNT+1;IF TOTCNT>23THEN BEGIN IF COUNTER>2XK THEN BEGIN COUNTER+0;C20000 145
+C2/2;C1+C1/2;IF C2<1THEN BEGIN SHANKS(N,X,XF,Y(CY0,*),FUNCTION,RKSNF,RK0000 146
SODR,RKC,EX,EA,ER,DX);GO TO FINISH;END;TOTCNT+0;FOR I+1STEP 1UNTIL N DO 0000 147
FOR J+1STEP 1UNTIL KM1 DO BEGIN CYL1+(CY0+16-J)MOD 16;CYL2+(CY0+16-(2XJ)0000 148
)MOD 16;Y(CYL1,IJ)+Y(CYL2,IJ);F(CYL1,IJ)+F(CYL2,IJ);END;GO TO DUBSRT;END;END0000 149
;END;COUNTER+COUNT+1;GO TO STRT;FINISH;FOR I+1STEP 1 UNTIL N DO YIVL0000 150
J+Y(CY0,IJ);END BUTCHER;PROCEDURE COWELL(N,XI,XF,Y,F,EA,ER,P,DX,RKSNF,RKS0000 151

```

```

ORDER,RKSCOEFF,Q,COWELLCOEFF,START,SHANKS,VALUE N,XI,XF,P,DX,RKSFN,RKS0000 152
RDER,Q;INTEGER N,RKSFN,RKSORDE,Q;REAL XI,XF,P,DX;REAL ARRAY Y,EA,ER,0000 153
RKSCOEFF,COWELLCOEFF,Q;PROCEDURE F,SHANKS;INTEGER PROCEDURE START;BEG0000 154
IN INTEGER C1,M,MM1,QP1,INDX,I1,I2,I3,I,J,K,CYI;INTEGER CORRECTCNT0000 155
;REAL INT,C2,DFACTOR,X,H,T1,T2,T3,T4,T5,T6;BOOLEAN DFLAG,PFLAG;REAL A0000 156
RRAY FHC0,Q,Q0,0,NJ,YMID1,YYP,YC,YM,CS,FP,HDM1F,HDM1FMID,EAV,ERV,EAVD,ERV0000 157
DIO,NJ,PCOEFF,CDOEFF,MCOEFF,Q;LABEL L0,L1,L2,L3,L4,L5,STARTER,DOUBLER0000 158
,ACCEPT,CORRECT,CLOSER,INT;:=XF-XI;C1:=1;LO:CI:=CI+CI;IF CI<Q TH0000 159
EN GO TO L0;IF DX<0 THEN DX:=DX;IF DX#0 THEN BEGIN L1:IF ABS(INT)/0000 160
C1>DX THEN BEGIN C1:=C1+CI;GO TO L1;END END;C2:=C1;M:=Q DIV 2;M0000 161
M1:=M-1;QP1:=Q+1;TQP1:=QP1+Q;DFACTOR:=2.0*(Q+3);INDX:=0;X 0000 162
:=XI;F(N,X,Y,FHC,Q,*)J;STARTER:CI:=CI1;:=START(N,XI,XF,C1,EA,ER,F,Q,X,Y0000 163
,FH,FH,YMID1,INDX,TQP1,I,P,RKSFN,RKSCOEFF))XCI;C2:=C2 XI1-Q;INDX:=(0000 164
INDX+Q)MOD TQP1;IF C2<M THEN GO TO CLOSER;DOUBLER:H:=INT /CI;FOR K0000 165
:=0 STEP 1 UNTIL Q DO BEGIN PCOEFF[K]:=COWELLCOEFF[K]*H;CCOEFF[K]:=COW0000 166
ELLCOEFF[I1]:=K+QP1)*H;MCOEFF[K]:=COWELLCOEFF[I1+QP1]*H;END;T1:=(C10000 167
*P)*X0.0;FOR J:=1 STEP 1 UNTIL N DO BEGIN EAV[J]:=(EAV[J]/T1)0000 168
/DFACTOR;ERV[J]:=(ERV[J]/T1)/DFACTOR;END;T1:=MCOEFF[C0];FOR J 0000 169
:=1 STEP 1 UNTIL N DO HMIF[J]:=YMID1[J]-FH[INDX,J]*T1;CYI:=INDX+TQP10000 170
;I3:=CYI-QP1;FOR K:=1 STEP 1 UNTIL M DO BEGIN I1:=(CYI-K)MOD TQP10000 171
;I2:=(CI3+K)MOD TQP1;T1:=MCOEFF[K]*H;T2:=MCOEFF[CQP1-K];FOR J:=1 0000 172
STEP 1 UNTIL N DO HMIF[J]:=HMIF[J]-FH[I1,J]*T1-FH[I2,J]*T2;END;FOR J0000 173
:=1 STEP 1 UNTIL N DO HM1FMID[J]:=HM1FMID[J];DFLAG:=FALSE;ACCEPT:FOR I0000 174
:=1 STEP 1 UNTIL M DO BEGIN CYI:=(INDX+1)MOD TQP1;TQP1;X:=0000 175
XF=(C2-I)*X;I1:=(CYI-I)MOD TQP1;T1:=PCOEFF[C0];T2:=CCOEFF[I];FOR 0000 176
J:=1 STEP 1 UNTIL N DO BEGIN YP[J]:=(T4:=(HDM1F[J]:=HDM1F[J]+H X(T3:=0000 177
FH[I1,J]))+T1 X(T3;CS[J]:=T4+T2 X(T3;END;I3:=CYI-QP1;FOR K:=2 STEP0000 178
1 UNTIL M DO BEGIN I1:=(CYI-K)MOD TQP1;I2:=(CI3+K)MOD TQP1;T1:=PC0000 179
DEFF[K-I];T2:=CCOEFF[K];T3:=PCOEFF[C-K];T4:=CCOEFF[CQP1-K];FOR J:=0000 180
1 STEP 1 UNTIL N DO BEGIN YP[J]:=YP[J]+T1 X(T5:=FH[I1,J])+T3 X(T6:=FH0000 181
I2,J);CS[J]:=CS[J]+T2 X(T5+T4 X(T6;END;I1:=(CYI-Q)MOD TQP1;I2:=0000 182
(CYI-QP1)MOD TQP1;T1:=PCOEFF[C-Q];T2:=CCOEFF[CQ];T3:=PCOEFF[CQ];FOR 0000 183
J:=1 STEP 1 UNTIL N DO BEGIN YP[J]:=YP[J]+T1 X(T4:=FH[I1,J])+T3 X(FH0000 184
,J);CS[J]:=CS[J]+T2 X(T4;END;T2:=CCOEFF[C0];CORRECTCNT:=1;CORRECT:F(N,0000 185
X,Y,FP);FOR J:=1 STEP 1 UNTIL N DO IF (T1:=ABS(C(T3:=(Y[CJ]:=CS[J]+T20000 186
XFP[J]))-YP[J])>EAV[J])THEN BEGIN IF T1>ERV[J]*ABS(T3)THEN BEGIN FOR J0000 187
:=J+1 STEP 1 UNTIL N DO YC[J]:=CS[J]+T2 XFP[J];F(N,X,YC,FP);FOR J:=1 0000 188
STEP 1 UNTIL N DO IF (T1:=ABS(C(T3:=(Y[CJ]:=CS[J]+T2 XFP[J]))-Y[CJ])>E0000 189
AVC[J])THEN BEGIN IF T1>ERV[J]*ABS(T3)THEN BEGIN FOR J:=J+1 STEP 1 UNTI0000 190
L N DO YP[J]:=CS[J]+T2 XFP[J];CORRECTCNT:=CORRECTCNT+2;IF CORRECTCNT 0000 191

```

```

>8 THEN BEGIN INDX :=(CYI -I)MOD TQPI ;GO TO L5 END ;GO TO CORRECT END E0000 192
ND ;F(N,X,YP,FH[INDX,*])PFLAG :=TRUE ;CORRECTCNT :=CORRECTCNT +1 ;GO TO0000 193
L2 END END ;F(N,X,YP,FH[INDX,*])PFLAG :=FALSE ;L2:END ;I1 :=(CYI -M)MOD0000 194
D TQPI ;I1 :=MCDEFF[M];FOR J :=1 STEP 1 UNTIL N DO YMC[J]:=HDM1FMID[J]+T10000 195
xFH[I1,J];I3 :=CYI -Q ;FOR K :=0 STEP 1 UNTIL MM1 DO BEGIN I1 :=(CYI -K)MOD 196
)MOD TQPI ;I2 :=(I3 +K)MOD TQPI ;I1 :=MCDEFF[K];T2 :=MCDEFF[IQ -K];FOR J 0000 197
:=1 STEP 1 UNTIL N DO YMC[J]:=YMC[J]+T1 xFH[I1,J]+T2 xFH[I2,J]END ;IF DFLA0000 198
G THEN BEGIN FOR J :=1 STEP 1 UNTIL N DO IF (T2 :=ABS((T3 :=Y[CJ]) -YMC[J]))0000 199
)>EAVD[J]THEN BEGIN IF T2 >ERVD[J]xABS(T3)THEN BEGIN IF T2 >EAV[J]THEN B0000 200
EGIN IF T2 >ERVD[J]xABS(T3)THEN GO TO L4 END ;GO TO L3 END END ;C2 :=C2 -0000 201
M ;C2 :=C2 /2.0 ;IF PFLAG THEN FOR J :=1 STEP 1 UNTIL N DO Y[CJ]:=Y[CJ]JEL0000 202
SE FOR J :=1 STEP 1 UNTIL N DO Y[CJ]:=Y[CJ];C1 :=C1 DIV 2 ;IF C2 <M THEN 0000 203
GO TO CLOSER ;INDX :=INDX +1 ;FOR K :=1 STEP 1 UNTIL Q DO BEGIN INDX :=(0000 204
INDX +1)MOD TQPI ;I1 :=(INDX +K)MOD TQPI ;FOR J :=1 STEP 1 UNTIL N DO FH0000 205
[I]NDX,JJ:=FH[I1,J]END ;GO TO DOUBLER END ;J :=0 ;L3:FOR J :=J +1 STEP 1 0000 206
UNTIL N DO IF (T2 :=ABS((T3 :=Y[CJ]) -YMC[J]))>EAV[J]THEN BEGIN IF T2 >ERV(0000 207
JJxABS(T3)THEN BEGIN L4:INDX :=(CYI -M)MOD TQPI ;L5:C1 :=C1 +C1 ;X :=XF 0000 208
=C2 xH ;C2 :=C2 +C2 ;GO TO STARTER END END ;C2 :=C2 -M ;IF C2 >M THEN BE0000 209
GIN IF PFLAG THEN FOR J :=1 STEP 1 UNTIL N DO BEGIN YMID[IJ]:=Y[CJ];Y[CJ]:0000 210
=Y[P[CJ]];HDM1FMID[IJ]:=HDM1FMID[IJ]END ELSE FOR J :=1 STEP 1 UNTIL N DO BEGIN Y0000 211
MID[IJ]:=Y[CJ];Y[CJ]:=Y[CJ];HDM1FMID[IJ]:=HDM1FMID[IJ]END ;DFLAG :=TRUE ;GO TO 0000 212
ACCEPT END ;IF PFLAG THEN FOR J :=1 STEP 1 UNTIL N DO Y[CJ]:=Y[P[CJ]ELSE F0000 213
R J :=1 STEP 1 UNTIL N DO Y[CJ]:=Y[CJ];CLOSER:=IF C2 >0 THEN SHANKS(N,X,XF0000 214
,Y,F,RKSFN,RKSORDER,RKSCOEFF,P,EA,ER,ABS(INT)/C1)END ;PROCEDURE SHANKS(0000 215
N,XI,XF,YV,F,M,ORDER,CF,P,EA,ER,DX)VALUE N,XI,XF,M,ORDER,P,DX;INTEGER N0000 216
,M,ORDER;REAL XI,XF,P,DX;REAL ARRAY YV,CF,EA,ER,I0;PROCEDURE F;BEGIN INT0000 217
EGER I,J,K,L,COUNT,DXD,DXH,DXT,EFACTUR,ERANGE,ES,GAMMA,X,M;BOOLEAN CFSW,DSW0000 218
AL BETA,DCOUNT,DXD,DXH,DXT,EFACTUR,ERANGE,ES,GAMMA,X,M;BOOLEAN CFSW,DSW0000 219
;REAL ARRAY CFDC0((M+3)xM=2),FV(08M=108N),GV,YC,YM,YP(08N);DEFINE CFH=C0000 220
FD#;LABEL L1,L2,EXIT;INTEGER STEPR,STEPS,M,M=1;STEPS+STEPR+0;DXD+DXT+XF=0000 221
XI;IF DXT=0 THEN GO TO EXIT;IF DX=0 THEN DX+DXD;COUNT+1;WHILE ABS(CDX)<ABS(0000 222
DXD)DO BEGIN COUNT+COUNT;COUNT;DXD+DXT;COUNT;END;COUNT2+COUNT+COUNT;DXH+0000 223
DXT/COUNT2;DCOUNT+COUNT;EFACT+1;FOR I+1STEP 1UNTIL ORDER DO EFACT+EFACT+0000 224
EFACT;ERANGE+0.125/EFACT;EFACT+4/EFACT;EFACTOR+COUNT+PxEFACT;DKTR+0;NCF10000 225
+(MxM+M)DIV 2+M+M;NCF+NCF1+1;CFSW+FALSE;FOR I+0STEP 1UNTIL NCF1 DO BEGIN0000 226
CFDC[IJ]+CFDC[IJ]xDXD;CFH[IJ]+NCF1+CFH[IJ]xDXH;END;X+XI;XM+XI+DXH;L1;DSW+TRUE;F(0000 227
N,X,YP,YV,GV);IF CFSW THEN L+NCF1 ELSE L+1;FOR I+1STEP 1UNTIL M DO BEGIN 10000 228
I+I-1;L+L+1;BETA+CFDC[L];FOR K+1STEP 1UNTIL N DO YP[KJ]+GV[KJ]xBETA+YV[KJ];F0000 229
OR J+1STEP 1UNTIL I1 DO BEGIN L+L+1;BETA+CFDC[LJ];FOR K+1STEP 1UNTIL N DO 0000 230
YP[KJ]+FV[IJ,KJ]xBETA+YP[KJ];END;L+L+1;F,N,CFDC[LJ]+X,YP,FV[IJ,KJ];END;L+L+1;GA0000 231

```

```

MMA+CFD[L];FOR K+1STEP 1UNTIL N DO YP[K]+GV[K]*GAMMA+YV[K];FOR I+1STEP 1000 232
UNTIL M DO BEGIN L+L+1;GAMMA+CFD[L];FOR K+1STEP 1UNTIL N DO YP[K]+FV[I,K]0000 233
I+1;L+L+1;BETA+CFH[L];FOR K+1STEP 1UNTIL M DO BEGIN I+1STEP 1UNTIL N DO YV[K];FOR I+1STEP 10000 234
I+1;L+L+1;BETA+CFH[L];FOR K+1STEP 1UNTIL N DO YM[K]+GV[K]*BETA+YV[K];FOR I+1STEP 100000 235
J+1STEP 1UNTIL I DO BEGIN L+L+1;BETA+CFH[L];FOR K+1STEP 1UNTIL N DO YM0000 236
[K]+FV[I,K]*BETA+YV[K];END;L+L+1;F(N,CFH[L]+X,YM,FV[I,*]);END;L+L+1;GAMM0000 237
A+CFH[L];FOR K+1STEP 1UNTIL N DO YM[K]+GV[K]*GAMMA+YV[K];FOR I+1STEP 100000 238
TIL M DO BEGIN L+L+1;GAMMA+CFH[L];FOR K+1STEP 1UNTIL N DO YM[K]+FV[I,K]*X0000 239
GAMMA+YV[K];END;F(N,X,YM,FV[0,*]);IF CFSW THEN L+1ELSE L+1;NCF1;FOR I+1S0000 240
TEP 1UNTIL M DO BEGIN I+1;FOR K+1STEP 1UNTIL N DO YC[K]+YV[K];FOR J+00000 241
STEP 1UNTIL I DO BEGIN L+L+1;BETA+CFH[L];FOR K+1STEP 1UNTIL N DO YC[K]+0000 242
FV[I,K]*BETA+YV[K];END;L+L+1;F(N,CFH[L]+X,YM,YC,FV[I,*]);END;FOR K+1STEP 10000 243
UNTIL N DO YC[K]+YV[K];FOR I+1STEP 1UNTIL M DO BEGIN L+L+1;GAMMA+CFH[L];0000 244
FOR K+1STEP 1UNTIL N DO YC[K]+FV[I,K]*GAMMA+YV[K];END;FOR K+1STEP 1UNTIL0000 245
N DO BEGIN ES+ABS(YC[K]-YP[K])*XFACTOR;IF ES#0THEN BEGIN IF ES>EALKTHE0000 246
N IF ES>ABS(YC[K])*XER[K]THEN BEGIN DSW+FALSE;STEP+STEP+1;COUNT+COUNT2;0000 247
COUNT+COUNT;COUNT+DCOUNT+DCOUNT;DXD+DXH;DXH+DXT/COUNT2;EFACTOR+CO000 248
COUNT+PXEFACT;IF CFSW THEN BEGIN FOR I+1STEP 1UNTIL NCF1 DO CFH[I]+NCFJ+CF0000 249
I+1;DXH;CFSW+FALSE;END ELSE BEGIN FOR I+1STEP 1UNTIL NCF1 DO CFH[I]+CFI10000 250
J+1;DXH;CFSW+TRUE;END;X+X+(COUNT+1-DCOUNT-DCOUNT)*DXH+XI;FOR K+1STEP 1UNTI0000 251
L N DO YP[K]+YV[K];GO TO L2;END;IF DSW THEN IF ES>EALKTXRANGE THEN IF E0000 252
S>ABS(YC[K])*XER[K]*XRANGE THEN DSW+FALSE;END;END;DCOUNT+DCOUNT-1;STEP+S0000 253
TEPS+1;FOR K+1STEP 1UNTIL N DO YV[K]+YV[K];IF DCOUNT#0THEN GO TO EXIT;X+0000 254
(COUNT-DCOUNT)*DXD+XI;IF DCOUNT<2THEN BEGIN IF DCOUNT=1THEN DSW+FALSE;IF0000 255
DCOUNT<1OR DSW THEN BEGIN IF DCOUNT>1THEN DKTR+DKTR+1;COUNT+DCOUNT+1;CO0000 256
UNT2+2;EFACTOR+EFACT;XI+X;DXD+DXT+XF=XI;DXH+DXD/2;XM+XI+DXH;CFSW+FALSE;F0000 257
OR I+1STEP 1UNTIL NCF1 DO BEGIN CFD[I]+CF[I]*DXD;CFH[I]+NCFJ+CFI1+DXH;EN0000 258
D;GO TO L1;END;END;IF DSW THEN BEGIN DKTR+DKTR+1;COUNT+COUNT;COUNT+CO0000 259
T DIV 2;DCOUNT+DCOUNT/2;DXH+DXD;DXD+DXT/COUNT;EFACTOR+COUNT+PXEFACT;IF CO000 260
FSW THEN BEGIN FOR I+1STEP 1UNTIL NCF1 DO CFD[I]+CF[I]*DXD;CFSW+FALSE;EN0000 261
D ELSE BEGIN FOR I+1STEP 1UNTIL NCF1 DO CFD[I]+NCFJ+CF[I]*DXD;CFSW+TRUE;E0000 262
NU;END;X+X+(COUNT+1-DCOUNT-DCOUNT)*DXH+XI;GO TO L1;EXIT;END;
INTEGER HSI;
FILE AB31HST DISK RANDOM "REMOTE"AB31S**"(1*90);
MONITOR INTOVR;
FORMAT FMINTOVR(" INTEGER OVERFLOW");
LABEL L39;
FORMAT F2I1(2I1+I10);
FORMAT F4E1(4E20+9+I9);
REAL TEMP;

```

```

DEFINE AC =ALLJ IF (TEMP :=ABS(Y [J]-Y1[J]))>EA[J]*2 AND TEMP >ER[J] 01022000
XABS(Y [J]+Y1[J]) THEN WRITE(PUN ,F4E1I,A,C,Y1[J],Y [J],J)#; 01023000
DEFINE SS =START,SHANKS#; 01024000
FORMAT FM5I10 (4(2I1,I8,X2),2(I1,I8,X2),2I10); 01025000
REAL A,C,DDX; 01026000
OWN ARRAY COEF[0:3,0:3,0:39]; 01027000
INTEGER ARRAY Q[2:73]; 01028000
OWN INTEGER KX,KY,KX33,KX41,KY33,KY41; 01029000
OWN INTEGER JS,K; 01030000
OWN INTEGER ARRAY AT,BJ[0:4]; 01031000
LIST LST(FOR J:=2 STEP 1 UNTIL 72 DO Q[J]); 01032000
DEFINE HEREBEFORE =AT[A]#,ALLJ=FOR J :=1 STEP 1 UNTIL N DO #,Y0BY 01033000
=ALLJ Y0[J]:=Y[J]#,SETY1ANDY =ALLJ 01034000
BEGIN 01035000
Y1[J]:=Y[J]; 01036000
Y[J]:=Y0[J] 01037000
END #; 01038000
LABEL L29,L1,L2,L3,L4,L5,L6,L7,M00,M01,M02,M03,M10,M11,M12,M13,M20,M21 01039000
,M22,M23,M30,M31,M32,M33; 01040000
SWITCH SW :=M00,M01,M02,M03,M10,M11,M12,M13,M20,M21,M22,M23,M30,M31 , 01041000
M32,M33; 01042000
SWITCH RETURN :=L1,L2,L3,L4,L5,L6,L7; 01043000
INTEGER L,ALF,ALF1,ALF2,BET,BET1,BET2,ALFBEST,BEST0,TALF,TALF1,TALF2 01044000
,TBET,TBET1,TBET2,BETBEST,BESTM ; 01045000
ARRAY Y0,Y1[0:N]; 01046000
PROCEDURE RECORDORDER(S,M,X,XT,Y,YT,Q); 01047000
VALUE M,X,XT,Y,YT; 01048000
INTEGER M,X,XT,Y,YT; 01049000
ARRAY Q[*]; 01050000
BEGIN 01051000
KX :=KY :=1 +8*X; 01052000
KX :=KX +2*X; 01053000
KY :=KY +2*Y; 01054000
IF XT < YT THEN 01055000
BEGIN 01056000
Q[KY+1]:=Q[KY+1]-YT; 01057000
Q[KX ]:=Q[KX ]+XT; 01058000
01059000
01060000
END; 01061000

```

```

IF YT < XT THEN
  BEGIN
    Q[KX+1] := Q[KX+1] - XT;
    Q[KY] := Q[KY] + YT;
  END;
END;
PROCEDURE RECORDMETHODS(X, XT, Y, YT, Q);
VALUE X, XT, Y, YT;
INTEGER X, XT, Y, YT;
ARRAY Q[*];
BEGIN
  KX33 := 33 + 2 * X;
  KY33 := 33 + 2 * Y;
  KX41 := 8 + KX33 + 8 * Y;
  KY41 := 8 + KY33 + 8 * X;
  IF XT < YT THEN
    BEGIN
      Q[KY33+1] := Q[KY33+1] - YT;
      Q[KX33] := Q[KX33] - XT;
      Q[KY41+1] := Q[KY41+1] - YT;
      Q[KX41] := Q[KX41] + XT;
    END;
  END;
  IF YT < XT THEN
    BEGIN
      Q[KX33+1] := Q[KX33+1] - XT;
      Q[KY33] := Q[KY33] + YT;
      Q[KX41+1] := Q[KX41+1] - XT;
      Q[KY41] := Q[KY41] + YT;
    END;
  END;
END;
PROCEDURE SELECTORDERS(M, X, Y, Q);
INTEGER M, X, Y;
ARRAY Q[*];

```

```

BEGIN
  J:=(Q[0]:(Q[0]x4093 +3000001)MOD 16777216 ) ;
  J :=J/24;
  FOR S :=0 STEP 1 UNTIL 3 DO
  BEGIN
    J :=(J+1)MOD 4;
    K :=1 +8xM +2xJ;
    AT[S]:=Q[K]+Q[K+1];
    BJ[S]:=J;
  END;
  X:=BJ[0];
  Y :=IF AT[1]≥AT[2]THEN IF AT[1]≥AT[3]THEN BJ[1]ELSE BJ[3]ELSE IF AT
[2]≥AT[3]THEN BJ[2]ELSE BJ[3];
  IF AT[0]<AT[1]THEN IF AT[0]<AT[2]THEN IF AT[0]<AT[3]THEN
  SELECTORDERS(M,X,Y,Q);
END;
PROCEDURE SELECTMETHODS(X,Y,Q);
INTEGER X,Y;
ARRAY Q[*];
BEGIN
  J:=(Q[0]:(Q[0]x4093 +3000001)MOD 16777216 ) ;
  J :=J/24;
  FOR S :=0 STEP 1 UNTIL 3 DO
  BEGIN
    J :=(J+1)MOD 4;
    K :=33 +2xJ;
    AT[S]:=Q[K]+Q[K+1];
    BJ[S]:=J;
  END;
  X:=BJ[0];
  Y :=IF AT[1]≥AT[2]THEN IF AT[1]≥AT[3]THEN BJ[1]ELSE BJ[3]ELSE IF AT
[2]≥AT[3]THEN BJ[2]ELSE BJ[3];
  IF AT[0]<AT[1]THEN IF AT[0]<AT[2]THEN IF AT[0]<AT[3]THEN
  SELECTMETHODS(X,Y,Q);
END;

```



```

YOBYY;
C:=A+DDX;
WRITE(PUN ,F2I1,ALF,ALF1,HST);
TALF1 :=TIME(2);
L:=1;
GO TO SW[4×ALF+ALF1+1];
L1:TALF1 :=TIME(2)-TALF1;
SETY1ANDY;
WRITE(PUN ,F2I1,ALF,ALF2);
TALF2 :=TIME(2);
L:=2;
GO TO SW[4×ALF+ALF2+1];
L2:TALF2 :=TIME(2)-TALF2;
AC;
A:=C;
YOBYY :=(Y[J]+Y1[J])/2;
C:=A+DDX;
WRITE(PUN ,F2I1,BET,BET1);
TBET1 :=TIME(2);
L:=3;
GO TO SW[4×BET+BET1+1];
L3:TBET1 :=TIME(2)-TBET1;
SETY1ANDY;
WRITE(PUN ,F2I1,BET,BET2);
TBET2 :=TIME(2);
L:=4;
GO TO SW[4×BET+BET2+1];
L4:TBET2 :=TIME(2)-TBET2;
AC;
A:=C;
YOBYY :=(Y[J]+Y1[J])/2;
C:=A+DDX×2;
ALFBEST :=IF TALF1≤TALF2 THEN ALF1 ELSE ALF2;
BETBEST :=IF TBET1≤TBET2 THEN BET1 ELSE BET2;
WRITE(PUN ,F2I1,ALF,ALFBEST);
TALF :=TIME(2);
L:=5;
GO TO SW[4×ALF+ALFBEST+1];
L5:TALF :=TIME(2)-TALF;
SETY1ANDY;

```

```

WRITE(PUN ,F2I1,BET,BETBEST);
TBET :=TIME(2);
L:=6;
GO TO SW[4xBET+BETBEST+1];
L6:TBET :=TIME(2)-TBET ;
AC;
A:=C;
YOBYY :=(Y[J]+Y1[J])/2;
C:=XF;
FOR J :=1 STEP 1 UNTIL 40 DO Q[J]:=Q[J]*0.980;
RECORDORDERS(ALF,ALF1,TALF1,ALF2,TALF2,Q);
RECORDORDERS(BET,BET1,TBET1,BET2,TBET2,Q);
RECORDMETHODS(ALF,TALF,BET,TBET,Q);
Q[-1]:=Q[-1]+1;
WRITE(A831HST[HST],75,Q[*]);
WRITE(PUN ,FM5I10,ALF,ALF1,TALF1,ALF,ALF2,TALF2,BET,BET1,TBET1,BET,
BET2,TBET2,ALF,TALF,BET,TBET,Q[-1],HST);
IF TALF <TBET THEN
BEGIN
    BESTM :=ALF;
    BESTO :=ALFBEST
END ELSE
BEGIN
    BESTM :=BET;
    BESTO :=BETBEST
END;
L :=7;
GO TO SW[4xBESTM +BESTO +1];
BEGIN
    GO TO L29;
M00:COMMENT;
ADAMS(N,A,C,Y,F,P,3,DX,EA,ER,COEF[0,0,*],,4,4,COEF[3,0,*],SS )§
GO TO L29;
M01:COMMENT;
ADAMS(N,A,C,Y,F,P,4,DX,EA,ER,COEF[0,1,*],,4,4,COEF[3,0,*],SS )§
GO TO L29;
M02:COMMENT;
ADAMS(N,A,C,Y,F,P,5,DX,EA,ER,COEF[0,2,*],,4,4,COEF[3,0,*],SS )§
GO TO L29;

```

```

01222000
01223000
01224000
01225000
01226000
01227000
01228000
01229000
01230000
01231000
01232000
01233000
01234000
01235000
01236000
01237000
01238000
01239000
01240000
01241000
01242000
01243000
01244000
01245000
01246000
01247000
01248000
01249000
01250000
01251000
01252000
01253000
01254000
01255000
01256000
01257000
01258000
01259000
01260000
01261000

```

```

M03:COMMENT;
ADAMS(N,A,C,Y,F,P,6,DX,EA,ER,COEF[0,3,*],5,5,COEF[3,1,*],SS );
GO TO L29;
M10:COMMENT;
BUTCHER(N,A,C,2,EA,ER,DX,COEF[1,0,*],F,P,COEF[3,0,*],SS,Y,4,4);
GO TO L29;
M11:COMMENT;
BUTCHER(N,A,C,3,EA,ER,DX,COEF[1,1,*],F,P,COEF[3,0,*],SS,Y,4,4);
GO TO L29;
M12:COMMENT;
BUTCHER(N,A,C,4,EA,ER,DX,COEF[1,2,*],F,P,COEF[3,0,*],SS,Y,4,4);
GO TO L29;
M13:COMMENT;
BUTCHER(N,A,C,4,EA,ER,DX,COEF[1,2,*],F,P,COEF[3,1,*],SS,Y,5,5);
GO TO L29;
M20:COMMENT;
COWELL(N,A,C,Y,F,EA,ER,P,DX,5,5,COEF[3,1,*],4,COEF[2,0,*],SS);
GO TO L29;
M21:COMMENT;
COWELL(N,A,C,Y,F,EA,ER,P,DX,5,5,COEF[3,1,*],6,COEF[2,1,*],SS);
GO TO L29;
M22:COMMENT;
COWELL(N,A,C,Y,F,EA,ER,P,DX,5,5,COEF[3,1,*],8,COEF[2,2,*],SS);
GO TO L29;
M23:COMMENT;
COWELL(N,A,C,Y,F,EA,ER,P,DX,6,6,COEF[3,2,*],8,COEF[2,2,*],SS);
GO TO L29;
M30:COMMENT;
SHANKS(N,A,C,Y,F,4,4,COEF[3,0,*],P,EA,ER,DX);
GO TO L29;
M31:COMMENT;
SHANKS(N,A,C,Y,F,5,5,COEF[3,1,*],P,EA,ER,DX);
GO TO L29;
M32:COMMENT;
SHANKS(N,A,C,Y,F,6,6,COEF[3,2,*],P,EA,ER,DX);
GO TO L29;
M33:COMMENT;
SHANKS(N,A,C,Y,F,7,7,COEF[3,3,*],P,EA,ER,DX);
GO TO L29;
L39:WRITE(PUN,FMIN(TOVR));
01262000
01263000
01264000
01265000
01266000
01267000
01268000
01269000
01270000
01271000
01272000
01273000
01274000
01275000
01276000
01277000
01278000
01279000
01280000
01281000
01282000
01283000
01284000
01285000
01286000
01287000
01288000
01289000
01290000
01291000
01292000
01293000
01294000
01295000
01296000
01297000
01298000
01299000
01300000
01301000

```

01302000
01303000
01304000
01305000
01306000
01307000
01308000

ALL J Y [J]:=Y0[J];
DIFEQINT(N,A,C,Y,F,P,EA,ER,DX);
L29:GO TO RETURN[L];
END;
L7:
END;

IV. AUXILIARY PROGRAMS

There are five auxiliary programs needed to use the integration procedure. There are several programs for setting up the needed disk files initially and programs for dumping the history files from the disk.

A. The Coefficient File

The file of the coefficients is created by a program called CREATE COEFFICIENT FILE. This program reads in the coefficients as rational numbers from cards and writes them in the disk file as double-precision real numbers in a form suitable for use by either the single or double-precision programs. The program listing and the coefficients in rational form are given at the end of this chapter.

B. Setting Up History Files

The history file is created by a program called CREATE HISTORY FILE. This program reads in a history file from cards and writes it on the disk file. There are two such programs, one for the single-precision history file and one for the double precision. The only difference is that the double-precision file is called "REMOTE" "A831**D" and the single precision file is called "REMOTE" "A831S**". At the end of this chapter, the single-precision program for creating the history file is listed together with a sample input history. The double-precision program would be identical except for the disk file declaration which would be "A831**D" instead of "A831S**".

C. Printing of History Files

The program called WRITE HISTORY FILE will print and punch on cards the history files "A831**D" or "A831S**", which are on disk. The format on the punched cards is that needed for input by program CREATE HISTORY FILE for creating these files. The program for dumping the single-precision history file is listed at the end of this chapter. The double-precision program differs from the single only by using the file name "A831**D" for "A831S**" in the disk file declaration.

```

CREATE HISTORY FILE

BEGIN
  FILE FLOUT 6(2,15);
  FORMAT FM(X5, A5, I10, I10/ ( 8I10));
  FILE HISTXXX (2, 10);
  ARRAY A(0:1,-2:72);
  INTEGER L, J, K;
  LIST LST(FOR L := -2 STEP 1 UNTIL 72 DO A[J, L]);
  FILE A831HST DISK SERIAL[1:12] "REMOTE" "A831S**" (1, 90, SAVE 360);
  WRITE(FLOUT[NO]);
  J := 0;
  FOR K := 1 STEP 1 UNTIL 12 DO
    BEGIN
      READ(HISTXXX, FM, LST);
      A[J, -2J] := TIME(0);
      WRITE(A831HST, 75, A[J, *J]);
    END;
  REWIND(A831HST);
  J := 1;
  FOR K := 1 STEP 1 UNTIL 12 DO
    BEGIN
      READ (A831HST, 75, A[J, *J]);
      WRITE(FLOUT, FM, LST);
    END;
  LOCK(A831HST, SAVE);
END.

01403000
01404000
01405000
01406000
01407000
01408000
01409000
01410000
01411000
01412000
01413000
01414000
01415000
01416000
01417000
01418000
01419000
01420000
01421000
01422000
01423000
01424000
01425000
01426000
01427000
01428000
01429000
01430000

```

```

WRITE HISTORY FILE

BEGIN
FILE PUNCH 0(2,10);
FILE FLOUT 6(2,15);
FORMAT FM(X5, A5, I10, I10/ ( 8I10));
ARRAY A(0:1,-2:72);
INTEGER L, J, K;
LIST LST(FOR L := -2 STEP 1 UNTIL 72 DO A[J, L]);
FILE A831HST DISK SERIAL "REMOTE" "A831S**" (1, 90, SAVE 360);
WRITE(FLOUT(N0));
J := 1;
FOR K := 1 STEP 1 UNTIL 12 DO
BEGIN
READ (A831HST, 75, A[J, *]);
WRITE(FLOUT, FM, LST);
WRITE(PUNCH, FM, LST);
END;
END.

```

01431000
01432000
01433000
01434000
01435000
01436000
01437000
01438000
01439000
01440000
01441000
01442000
01443000
01444000
01445000
01446000
01447000
01448000
01449000

SAMPLE OUTPUT FROM WRITE HISTORY FILE AND INPUT TO CREARE HISTORY FILE

67167	52	9184675	-300	505	-247	0	-137
46	-317	270	-257	0	-180	0	-2339
917	-24	236	-291	25	-203	0	-351
559	0	0	-559	109	-416	188	-445
852	-434	226	-616	0	-611	1523	-78
70	-818	25	0	0	0	975	-149
0	0	0	0	0	-69	566	0
0	0	0	0	0	0	712	0
0	0	58	-973	0	-870	0	0
129	-1193	0					
67167	48	9253435	-248	611	-329	267	-327
0	-142	46	-94	46	-466	425	-368
292	-456	97					

571	0	-286	25	-344	0	-281
305	-564	-471	0	-387	190	-587
25	-1248	-723	45	-807	1846	-131
0	0	0	0	0	1212	-58
0	0	0	0	-72	588	-77
0	0	0	0	0	710	-123
41	-1574	-1053	96	-1087	0	0
67167	9	0	0	-212	350	-442
305	-200	-190	180	-409	360	0
0	0	-529	813	-691	0	-286
0	-228	0	134	-269	0	-169
233	0	-930	1770	-1544	214	-936
1050	-550	0	569	-1336	0	0
0	0	0	825	0	0	0
0	0	-1048	0	0	227	-1021
1072	-620	0	584	-250	0	0
0	0	0	0	0	0	0
67167	1	0	0	-179	0	0
0	0	0	0	-134	97	0
0	0	0	0	0	0	0
0	0	-187	156	0	0	0
0	0	0	0	0	0	0
0	0	0	156	0	0	0
0	0	-187	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
67167	68	-525	858	-497	219	-362
121	-285	-1375	1245	-1435	434	-89
48	-26	-198	82	-207	0	-159
305	-68	-8904	450	-524	962	-1078
352	-668	-973	347	-1632	1732	-897
738	-906	0	0	0	1038	-1526
0	0	0	1127	0	313	0
0	0	-1470	0	0	1687	-84
0	-1372	-965	78	-3646	0	0
1328	46	0	174	-172	556	-353
67167	-289	-429	25	-1352	894	-143
1914	-3618	-1832	0	0	0	0

641	-68	0	-378	232	-354	0	-320
297	-494	998	-486	230	-609	25	-417
73	-723	0	-716	74	-785	1351	-128
0	0	0	0	0	0	820	-125
0	0	0	0	67	0	355	0
0	0	0	-120	0	0	812	-110
115	-1045	0	-955	104	-1195	0	0
67167	202	16694821	-426	662	-618	0	-561
1068	-457	0	-2041	0	-1753	0	-2064
4720	-291	156	-4709	25	-4583	0	-4552
5617	0	0	-550	25	-431	0	-464
1053	0	0	-4739	6337	-4768	0	-2000
641	-2454	3618	0	10573	-5036	0	0
0	0	0	0	10017	-6490	0	0
4388	-14241	5908	-15101	0	0	0	-10834
0	0	0	0	7138	0	0	0
67167	4	14195761	-126	215	0	0	-159
0	0	0	0	0	-607	0	0
0	0	290	-389	666	-165	0	-459
78	0	0	0	143	0	0	-182
0	0	0	-588	776	0	0	-189
0	-411	0	0	293	0	0	0
0	0	0	0	367	0	0	0
0	0	0	-625	0	0	0	-197
0	-415	0	0	146	0	0	0
67167	48	9222461	-667	57	-572	907	-332
123	-96	90	-71	0	-246	0	-76
68	-296	357	-155	1295	-199	75	-117
71	-109	0	-628	1509	-833	289	-555
267	-1128	150	-1356	62	-1320	1694	-114
41	-781	25	0	0	0	853	-44
0	0	0	0	0	-67	831	0
0	0	0	0	0	0	941	-98
0	0	62	-1975	87	-1855	0	0
41	-1223	0	-299	184	-97	48	-235
67167	47	14413612	-58	0	-70	505	-1558
0	-1210	1037					
0	-575	1544					

176	-424	0	-1269	791	-282	48	-290
56	-861	1020	-981	760	-630	730	-1064
88	-982	0	-1266	62	-962	1556	-129
0	0	0	0	0	0	1047	-157
0	0	0	0	71	0	271	0
0	0	0	-560	0	0	1005	-46
142	-1478	0	-1490	43	-1576	0	0
67167	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
67167	69	1479195	-5179	6762	-1203	886	-1382
0	-1189	295	-2352	2964	-1204	210	-2400
770	-1183	108	-945	475	-1094	530	-718
1410	-1353	584	-10497	25060	-995	51	-1835
276	-36028	661	-29880	1738	-2604	4445	-7276
4378	-2539	3199	-871	944	-1005	1870	-4087
0	0	0	0	73	-164	3584	-4296
828	0	0	-93	0	0	2511	-1542
953	-1413	132	-47388	1539	-3224	0	0
3510	-1996	4217					

```

CREATE COFFICIENT FILE
BEGIN
  FILE IN CARD(2,10) ;
  FILE OUT PRINT 6(2,15) ;
  SAVE FILE TAPE831 DISK SERIAL[1:76] "REMDTE"TAPE831"(2,90, SAVE 360)
  ;
  INTEGER I,J ;
  SAVE REAL ARRAY A,B[0:88] ;
  LABEL LAST ;
  SAVE INTEGER ARRAY BUFF[0:14],RATNO[0:5] ;
  PROCEDURE RREAD(AH,AL) ;
  REAL AH,AL ;
  BEGIN
    STREAM PROCEDURE CONCARD(S,D) ;
    BEGIN
      LOCAL M,N ;
      SI := S ;
      SI := SI + 24 ;
      DI := S ;
      24(IF SC = " " THEN DS := 1 NUM ELSE JUMP OUT) ;
      IF SC = "" THEN
        BEGIN
          TALLY := 1 ;
          SI := SI + 1
        END ELSE
        BEGIN
          TALLY := 0 ;
          DI := DI - 1
        END ;
        M := TALLY ;
        24(IF SC = "/" THEN DS := 1 CHR ELSE JUMP OUT) ;
        SI := SI + 1 ;
        TALLY := 0 ;
        24(IF SC ≥ "0" THEN
          BEGIN
            SI := SI + 1 ;
            TALLY := TALLY + 1
          END ELSE DS := 1 NUM) ;
    END
  END

```

```

01310000
01311000
01312000
01313000
01314000
01315000
01316000
01317000
01318000
01319000
01320000
01321000
01322000
01323000
01324000
01325000
01326000
01327000
01328000
01329000
01330000
01331000
01332000
01333000
01334000
01335000
01336000
01337000
01338000
01339000
01340000
01341000
01342000
01343000
01344000
01345000
01346000
01347000
01348000

```



```

01389000
01390000
01391000
01392000
01393000
01394000
01395000
01396000
01397000
01398000
01399000
01400000
01401000
01402000

```

```

WRITE(TAPE831,I,A[*]) ;
WRITE(PRINT[DBL]) ;
WRITE(TAPE831,I,B[*])
END ;
FOR I := 13,19,26,34,53,64,89 DO
BEGIN
  FOR J := 0 STEP 1 UNTIL I - 1 DO RREAD(A[J],B[J]) ;
  WRITE(TAPE831,I,A[*]) ;
  WRITE(PRINT[DBL]) ;
  WRITE(TAPE831,I,B[*])
END ;
LOCK(TAPE831,RELEASE) ;
LAST:
END .

```

DATA INPUT TO CREATE COEFFICIENT FILE

BETA (2, 2) =	5/12
BETA (2, 1) =	-4/3
BETA (2, 0) =	23/12
BETA* (3, 3) =	1/24
BETA* (3, 2) =	-5/24
BETA* (3, 1) =	19/24
BETA* (3, 0) =	3/8
Q = 2 RATIO (4) =	270/19
BETA (3, 3) =	-3/8
BETA (3, 2) =	37/24
BETA (3, 1) =	-59/24
BETA (3, 0) =	55/24
BETA* (4, 4) =	-19/720
BETA* (4, 3) =	53/360
BETA* (4, 2) =	-11/30
BETA* (4, 1) =	323/360
BETA* (4, 0) =	251/720
Q = 3 RATIO (5) =	502/27
BETA (4, 4) =	251/720
BETA (4, 3) =	-637/360
BETA (4, 2) =	109/30
BETA (4, 1) =	-1387/360
BETA (4, 0) =	1901/720
BETA* (5, 5) =	3/160
BETA* (5, 4) =	-173/1440
BETA* (5, 3) =	241/720
BETA* (5, 2) =	-133/240
BETA* (5, 1) =	1427/1440
BETA* (5, 0) =	95/288
Q = 4 RATIO (6) =	19950/863
BETA (5, 5) =	-95/288
BETA (5, 4) =	959/480
BETA (5, 3) =	-3649/720
BETA (5, 2) =	4991/720
BETA (5, 1) =	-2641/480
BETA (5, 0) =	4277/1440
BETA* (6, 6) =	-863/60480
BETA* (6, 5) =	263/2520

BETA* (6, 4) =	-6737/20160
BETA* (6, 3) =	586/945
BETA* (6, 2) =	-15487/20160
BETA* (6, 1) =	2713/2520
BETA* (6, 0) =	19087/60480
Q = 5 RATIO (7) =	38174/1375
BETA* (6, 6) =	19087/60480
BETA* (6, 5) =	-5603/2520
BETA* (6, 4) =	135713/20160
BETA* (6, 3) =	-10754/945
BETA* (6, 2) =	235183/20160
BETA* (6, 1) =	-18637/2520
BETA* (6, 0) =	198721/60480
BETA* (7, 7) =	275/24192
BETA* (7, 6) =	-11351/120960
BETA* (7, 5) =	1537/4480
BETA* (7, 4) =	-88547/120960
BETA* (7, 3) =	123133/120960
BETA* (7, 2) =	-4511/4480
BETA* (7, 1) =	139849/120960
BETA* (7, 0) =	5257/17280
Q = 6 RATIO (8) =	1103970/33953
BETA* (7, 7) =	-5257/17280
BETA* (7, 6) =	32863/13440
BETA* (7, 5) =	-115747/13440
BETA* (7, 4) =	2102243/120960
BETA* (7, 3) =	-296053/13440
BETA* (7, 2) =	242653/13440
BETA* (7, 1) =	-1152169/120960
BETA* (7, 0) =	16083/4480
BETA* (8, 8) =	-33953/3628800
BETA* (8, 7) =	156437/1814400
BETA* (8, 6) =	-645607/1814400
BETA* (8, 5) =	1573169/1814400
BETA* (8, 4) =	-31457/22680
BETA* (8, 3) =	2797679/1814400
BETA* (8, 2) =	-2302297/1814400
BETA* (8, 1) =	2233547/1814400
BETA* (8, 0) =	1070017/3628800
Q = 7 RATIO (9) =	2140034/57281

BETA (8, 8)	=	1070017/3628800
BETA (8, 7)	=	-4832053/1814400
BETA (8, 6)	=	19416743/1814400
BETA (8, 5)	=	-45586321/1814400
BETA (8, 4)	=	862303/22680
BETA (8, 3)	=	-69927631/1814400
BETA (8, 2)	=	47738393/1814400
BETA (8, 1)	=	-21562603/1814400
BETA (8, 0)	=	14097247/3628800
BETA* (9, 9)	=	8183/1036800
BETA* (9, 8)	=	-116687/1451520
BETA* (9, 7)	=	335983/907200
BETA* (9, 6)	=	-462127/453600
BETA* (9, 5)	=	6755041/3628800
BETA* (9, 4)	=	-8641623/3628800
BETA* (9, 3)	=	200029/90720
BETA* (9, 2)	=	-1408913/907200
BETA* (9, 1)	=	9449717/7257600
BETA* (9, 0)	=	25713/89600
Q = 8 RATIO (10)	=	137461698/3250433
BETA (9, 9)	=	-25713/89600
BETA (9, 8)	=	20884811/7257600
BETA (9, 7)	=	-2357683/181440
BETA (9, 6)	=	15788639/453600
BETA (9, 5)	=	-222386081/3628800
BETA (9, 4)	=	269181919/3628800
BETA (9, 3)	=	-28416361/453600
BETA (9, 2)	=	6648317/181440
BETA (9, 1)	=	-104995189/7257600
BETA (9, 0)	=	4325321/1036800
BETA* (10, 10)	=	-3250433/479001600
BETA* (10, 9)	=	9071219/119750400
BETA* (10, 8)	=	-12318413/31933440
BETA* (10, 7)	=	23643791/19958400
BETA* (10, 6)	=	-21677723/8870400
BETA* (10, 5)	=	2227571/623700
BETA* (10, 4)	=	-33765029/8870400
BETA* (10, 3)	=	12051709/3991680
BETA* (10, 2)	=	-296725183/159667200
BETA* (10, 1)	=	164046413/119750400

BETA* (10, 0) =
 Q = 9 RATIO (11) =
 BETA (10, 10) =
 BETA (10, 9) =
 BETA (10, 8) =
 BETA (10, 7) =
 BETA (10, 6) =
 BETA (10, 5) =
 BETA (10, 4) =
 BETA (10, 3) =
 BETA (10, 2) =
 BETA (10, 1) =
 BETA (10, 0) =
 BETA* (11, 11) =
 BETA* (11, 10) =
 BETA* (11, 9) =
 BETA* (11, 8) =
 BETA* (11, 7) =
 BETA* (11, 6) =
 BETA* (11, 5) =
 BETA* (11, 4) =
 BETA* (11, 3) =
 BETA* (11, 2) =
 BETA* (11, 1) =
 BETA* (11, 0) =
 Q = 10 RATIO (12) =
 BETA (11, 11) =
 BETA (11, 10) =
 BETA (11, 9) =
 BETA (11, 8) =
 BETA (11, 7) =
 BETA (11, 6) =
 BETA (11, 5) =
 BETA (11, 4) =
 BETA (11, 3) =
 BETA (11, 2) =
 BETA (11, 1) =
 BETA (11, 0) =
 BETA* (12, 12) =
 BETA* (12, 11) =

26842253/95800320
 53684506/1135053
 26842253/95800320
 -52841941/17107200
 2472634817/159667200
 -186080291/3991680
 2492064913/26611200
 -82260679/623700
 3539798831/26611200
 -1921376209/19958400
 1572737587/31933440
 -2067948781/119750400
 2132509567/479001600
 4671/788480
 -68928781/958003200
 384709327/958003200
 -87064741/63866880
 501289903/159667200
 -91910491/17740800
 1007253581/159667200
 -102212233/17740800
 36465037/9123840
 -99642413/45619200
 1374799219/958003200
 4777223/17418240
 717300033450/13695779093
 -4777223/17418240
 30082309/9123840
 -17410248271/958003200
 923636629/15206400
 -625551749/4561920
 35183928883/159667200
 -41290273229/159667200
 35689892561/159667200
 -15064372973/106444800
 12326645437/191600640
 -6477936721/319334400
 4527766399/958003200
 -13695779093/2615348736000
 2724891251/39626496000

-30336027563/72648576000
 406332786317/261534873600
 -229882484333/58118860800
 529394045911/72648576000
 -4874320027/486486000
 84400835489/8072064000
 -485500845331/58118860800
 1346577425651/261534873600
 -551368413119/217945728000
 6595204069/4402944000
 703604254357/2615348736000
 1407208508714/24466579093
 703604254357/2615348736000
 -169639834921/48432384000
 4588414555201/217945728000
 -20232291373837/261534873600
 2253957198793/11623772160
 -2826800577631/8072064000
 228133014533/486486000
 -34266367915049/72648576000
 20730767690131/58118860800
 -10498491598103/52306974720
 5963794194517/72648576000
 -931781102989/39626496000
 13064406523627/2615348736000
 2224234463/475517952000
 -69091417279/1046139494400
 1636420501/3773952000
 -4590817802567/2615348736000
 5124051955567/1046139494400
 -5797545653629/581188608000
 1335017017153/87178291200
 -786611554491/435891456000
 9575580965507/581188608000
 -1748248003751/149448499200
 16964495066809/2615348736000
 -168235945379/58118860800
 741197087471/475517952000
 106364763817/402361344000
 8296451577726/1322822840127

BETA* (12, 10) =
 BETA* (12, 9) =
 BETA* (12, 8) =
 BETA* (12, 7) =
 BETA* (12, 6) =
 BETA* (12, 5) =
 BETA* (12, 4) =
 BETA* (12, 3) =
 BETA* (12, 2) =
 BETA* (12, 1) =
 BETA* (12, 0) =
 Q = 11 RATIO (13) =
 BETA (12, 12) =
 BETA (12, 11) =
 BETA (12, 10) =
 BETA (12, 9) =
 BETA (12, 8) =
 BETA (12, 7) =
 BETA (12, 6) =
 BETA (12, 5) =
 BETA (12, 4) =
 BETA (12, 3) =
 BETA (12, 2) =
 BETA (12, 1) =
 BETA (12, 0) =
 BETA* (13, 13) =
 BETA* (13, 12) =
 BETA* (13, 11) =
 BETA* (13, 10) =
 BETA* (13, 9) =
 BETA* (13, 8) =
 BETA* (13, 7) =
 BETA* (13, 6) =
 BETA* (13, 5) =
 BETA* (13, 4) =
 BETA* (13, 3) =
 BETA* (13, 2) =
 BETA* (13, 1) =
 BETA* (13, 0) =
 Q = 12 RATIO (14) =

BETA (13,13) =
 BETA (13,12) =
 BETA (13,11) =
 BETA (13,10) =
 BETA (13, 9) =
 BETA (13, 8) =
 BETA (13, 7) =
 BETA (13, 6) =
 BETA (13, 5) =
 BETA (13, 4) =
 BETA (13, 3) =
 BETA (13, 2) =
 BETA (13, 1) =
 BETA (13, 0) =
 BETA* (14,14) =
 BETA* (14,13) =
 BETA* (14,12) =
 BETA* (14,11) =
 BETA* (14,10) =
 BETA* (14, 9) =
 BETA* (14, 8) =
 BETA* (14, 7) =
 BETA* (14, 6) =
 BETA* (14, 5) =
 BETA* (14, 4) =
 BETA* (14, 3) =
 BETA* (14, 2) =
 BETA* (14, 1) =
 BETA* (14, 0) =
 Q =13
 BETA (14,14) =
 BETA (14,13) =
 BETA (14,12) =
 BETA (14,11) =
 BETA (14,10) =
 BETA (14, 9) =
 BETA (14, 8) =
 BETA (14, 7) =
 BETA (14, 6) =
 BETA (14, 5) =

-106364763817/402361344000
 6460951197929/1743565824000
 -21029162113651/871782912000
 7222659159949/74724249600
 -10320787460413/38745907200
 310429955875453/581188608000
 -350379327127877/435891456000
 134046425652457/145297152000
 -31457535950413/38745907200
 570885914358161/1046139494400
 -34412222659093/124540416000
 89541175419277/871782912000
 -140970750679621/5230697472000
 905730205/172204032
 -132282840127/31384184832000
 124922452271/1961511552000
 -14110480969927/31384184832000
 137855863153/70053984000
 -187504936597931/31384184832000
 26159487787579/1961511552000
 -236770944732449/10461394944000
 38029005269/1277025750
 -321201800274911/10461394944000
 48869476129477/1961511552000
 -71363886250691/4483454976000
 3933201478249/490377888000
 -102885148956217/31384184832000
 3173185470929/1961511552000
 1166309819657/4483454976000
 18660957114512/274523709512
 1166309819657/4483454976000
 -696561442637/178319232000
 859236476684231/31384184832000
 -58262613384023/490377888000
 1600835679073597/4483454976000
 -1544031478475483/1961511552000
 13760072112094753/10461394944000
 -2166615342637/1277025750
 17823675553313503/10461394944000
 -2614079370781733/1961511552000

BETA (14, 4)	=	25298910337081429/31384184832000
BETA (14, 3)	=	-25990262345039/70053984000
BETA (14, 2)	=	3966421670215481/31384184832000
BETA (14, 1)	=	-60007679150257/1961511552000
BETA (14, 0)	=	13325653738373/2414168064000
BETA* (15, 15)	=	2639651053/689762304000
BETA* (15, 14)	=	-3867689367599/62768369664000
BETA* (15, 13)	=	29219384284087/62768369664000
BETA* (15, 12)	=	-137515713789319/62768369664000
BETA* (15, 11)	=	64486158419069/8966909952000
BETA* (15, 10)	=	-1096355235402331/62768369664000
BETA* (15, 9)	=	679781959848881/20922789888000
BETA* (15, 8)	=	-988788576755233/20922789888000
BETA* (15, 7)	=	1138313909617631/20922789888000
BETA* (15, 6)	=	-3129453071993581/62768369664000
BETA* (15, 5)	=	2285168598349733/62768369664000
BETA* (15, 4)	=	-189568380436867/8966909952000
BETA* (15, 3)	=	612744541065337/62768369664000
BETA* (15, 2)	=	-2099287611259/5706215424000
BETA* (15, 1)	=	105145058757073/62768369664000
BETA* (15, 0)	=	25221445/98402304
Q = 14 RATIO (16)	=	8204945906981250/111956703448001
BETA (15, 15)	=	-25221445/98402304
BETA (15, 14)	=	36807182273689/8966909952000
BETA (15, 13)	=	-1934443196892599/62768369664000
BETA (15, 12)	=	9038571752734087/62768369664000
BETA (15, 11)	=	-267435537386529/5706215424000
BETA (15, 10)	=	10103478797549069/8966909952000
BETA (15, 9)	=	-129930094104237331/62768369664000
BETA (15, 8)	=	62029181421198881/20922789888000
BETA (15, 7)	=	-70006862970773983/20922789888000
BETA (15, 6)	=	62487713370967631/20922789888000
BETA (15, 5)	=	-131963191940828581/62768369664000
BETA (15, 4)	=	72558117072259733/62768369664000
BETA (15, 3)	=	-4372481980074367/8966909952000
BETA (15, 2)	=	740161300731949/4828336128000
BETA (15, 1)	=	-2161567671248849/62768369664000
BETA (15, 0)	=	362555126427073/62768369664000
BETA* (16, 16)	=	-111956703448001/32011868528640000
BETA* (16, 15)	=	956906730268873/16005934264320000

BETA*	(16,14)	=	-171192511013729/355687428096000
BETA*	(16,13)	=	596904922428961/246245142528000
BETA*	(16,12)	=	-27389421430791451/3201186852864000
BETA*	(16,11)	=	5708273541404323/254062448640000
BETA*	(16,10)	=	-72784522563390409/16005934264320000
BETA*	(16, 9)	=	232085108601391937/3201186852864000
BETA*	(16, 8)	=	-42733352080603/463134672000
BETA*	(16, 7)	=	302240496916010687/3201186852864000
BETA*	(16, 6)	=	-1246285173964159159/16005934264320000
BETA*	(16, 5)	=	91914603656624011/1778437140480000
BETA*	(16, 4)	=	-12578861691928243/457312407552000
BETA*	(16, 3)	=	37519546987420243/3201186852864000
BETA*	(16, 2)	=	-1458231199032479/355687428096000
BETA*	(16, 1)	=	27707643610637623/16005934264320000
BETA*	(16, 0)	=	8092989203533249/32011868528640000
Q = 15	RATIO (17)	=	16185978407066498/205804074990625
BETA	(16,16)	=	8092989203533249/32011868528640000
BETA	(16,15)	=	-68846386581756617/16005934264320000
BETA	(16,14)	=	942359269351333/27360571392000
BETA	(16,13)	=	-551863998439384493/3201186852864000
BETA	(16,12)	=	386778238886497951/640237370572800
BETA	(16,11)	=	-2797406189209536629/1778437140480000
BETA	(16,10)	=	7205576917796031023/2286562037760000
BETA	(16, 9)	=	-324179886697104913/65330343936000
BETA	(16, 8)	=	2879939505554213/463134672000
BETA	(16, 7)	=	-3993885936674091251/640237370572800
BETA	(16, 6)	=	8020742949973736671/16005934264320000
BETA	(16, 5)	=	-5702855818380878219/1778437140480000
BETA	(16, 4)	=	5173388005728297701/3201186852864000
BETA	(16, 3)	=	-22133864200927593/35177877504000
BETA	(16, 2)	=	2612634723678583/14227497123840
BETA	(16, 1)	=	-55994879072429317/1455084933120000
BETA	(16, 0)	=	14845854129333883/2462451425280000
BETA*	(17,17)	=	50186465/15613165568
BETA*	(17,16)	=	-3722582669836627/64023737057280000
BETA*	(17,15)	=	1136320750876589/2286562037760000
BETA*	(17,14)	=	-72974966880383/27360571392000
BETA*	(17,13)	=	8062612208040217/800296713216000
BETA*	(17,12)	=	-45532601008155413/1600593426432000
BETA*	(17,11)	=	15815540301010573/254062448640000

BETA* (17, 10)	=	-1728464634834409159/16005934264320000
BETA* (17, 9)	=	964479921803293249/6402373705728000
BETA* (17, 8)	=	-40409465324546861/237124952064000
BETA* (17, 7)	=	502364378756214437/3201186852864000
BETA* (17, 6)	=	-1883042979819352909/16005934264320000
BETA* (17, 5)	=	63645018657622943/889218570240000
BETA* (17, 4)	=	-4019382738717217/114328101888000
BETA* (17, 3)	=	44516885513301493/3201186852864000
BETA* (17, 2)	=	-146702510065339/32335220736000
BETA* (17, 1)	=	11432924370549117/64023737057280000
BETA* (17, 0)	=	85455477715379/342372925440000
Q = 16 RA* ID (18)	=	12752179117555146654/151711881512390095
BETA (17, 17)	=	-85455477715379/342372925440000
BETA (17, 16)	=	2460247368070567/547211427840000
BETA (17, 15)	=	-612172313896136299/16005934264320000
BETA (17, 14)	=	653581961828485643/3201186852864000
BETA (17, 13)	=	-4543527303777247/5928123801600
BETA (17, 12)	=	17195392832483362153/8002967132160000
BETA (17, 11)	=	-74619315088494380723/16005934264320000
BETA (17, 10)	=	14237182892280945743/1778437140480000
BETA (17, 9)	=	-70617432699294428737/6402373705728000
BETA (17, 8)	=	3146403501110383511/256094948229120
BETA (17, 7)	=	-19726972891423175089/1778437140480000
BETA (17, 6)	=	129650088885345917773/16005934264320000
BETA (17, 5)	=	-38023516029116089751/8002967132160000
BETA (17, 4)	=	65509525475265061/29640619008000
BETA (17, 3)	=	-511501877919758129/640237370572800
BETA (17, 2)	=	497505713064683651/2286562037760000
BETA (17, 1)	=	-3947240465864473/92386344960000
BETA (17, 0)	=	57424625956493833/9146248151040000
BETA* (18, 18)	=	-2334028946344463/786014494949376000
BETA* (18, 17)	=	16063586213927447/283838567620608000
BETA* (18, 16)	=	-8727512947308437627/17030314057236480000
BETA* (18, 15)	=	6216118590489229589/2128789257154560000
BETA* (18, 14)	=	-256626484009824989/21833735970616000
BETA* (18, 13)	=	2150219719237435461/60822550204416000
BETA* (18, 12)	=	-23052478094030507/275839229952000
BETA* (18, 11)	=	111229455669198952379/709596419031520000
BETA* (18, 10)	=	-2025977937601766635423/8515157028618240000
BETA* (18, 9)	=	735975656988412429/2494674910728000

BETA* (18, 8) =	-170502576778289249423/567677135241216000
BETA* (18, 7) =	21409690670332474663/85151570286182400
BETA* (18, 6) =	-105083959047373621537/60822550204160000
BETA* (18, 5) =	127722520943891521/1316505415680000
BETA* (18, 4) =	-37673668595097727061/851515702861824000
BETA* (18, 3) =	463492437836890081/28383856762060800
BETA* (18, 2) =	-5666838930556309291/1135354270482432000
BETA* (18, 1) =	7830462528683744423/4257578514309120000
BETA* (18, 0) =	12600467236042756559/51090942171709440000
Q = 17	25200934472085513118/281550972898020815
RATIO (19) =	1/1
K = 1 CAP A (1) =	1/2
K = 1 CAP B (1) =	1/1
K = 1 LC A (1) =	1/1
K = 1 LC B (1) =	-1/1
K = 1 ALPHA (1) =	1/1
K = 1 BETA (1) =	1/6
K = 1 LC B =	2/1
K = 1 BETA =	4/6
K = 1 BETA (0) =	1/6
K = 2 CAP A (1) =	0/1
K = 2 CAP B (1) =	9/8
K = 2 LC A (1) =	28/5
K = 2 LC B (1) =	-60/15
K = 2 ALPHA (1) =	32/31
K = 2 BETA (1) =	12/93
K = 2 CAP A (2) =	1/1
K = 2 CAP B (2) =	3/8
K = 2 LC A (2) =	-23/5
K = 2 LC B (2) =	-26/15
K = 2 ALPHA (2) =	-1/31
K = 2 BETA (2) =	-1/93
K = 2 LC B =	32/15
K = 2 BETA =	64/93
K = 2 BETA (0) =	15/93
K = 3 CAP A (1) =	-225/128
K = 3 CAP B (1) =	225/128
K = 3 LC A (1) =	540/31
K = 3 LC B (1) =	-1395/155
K = 3 ALPHA (1) =	783/617
K = 3 BETA (1) =	-135/3085

K	=	3	CAP	A	(2)	=	200/128
K	=	3	CAP	B	(2)	=	300/128
K	=	3	LC	A	(2)	=	-297/31
K	=	3	LC	B	(2)	=	-2130/155
K	=	3	ALPHA	(2)	=	-135/617	
K	=	3	BETA	(2)	=	-495/3085	
K	=	3	CAP	A	(3)	=	153/128
K	=	3	CAP	B	(3)	=	45/128
K	=	3	LC	A	(3)	=	-212/31
K	=	3	LC	B	(3)	=	-309/155
K	=	3	ALPHA	(3)	=	-31/617	
K	=	3	BETA	(3)	=	-39/3085	
K	=	3	LC	B	=	384/155	
K	=	3	BETA	=	2304/3085		
K	=	3	BETA	(0)	=	465/3085	
K	=	4	CAP	A	(1)	=	-629200/59049
K	=	4	CAP	B	(1)	=	96800/19683
K	=	4	LC	A	(1)	=	3796/61
K	=	4	LC	B	(1)	=	-75240/3355
K	=	4	ALPHA	(1)	=	2248/3943	
K	=	4	BETA	(1)	=	129360/216865	
K	=	4	CAP	A	(2)	=	-418176/59049
K	=	4	CAP	B	(2)	=	348480/19683
K	=	4	LC	A	(2)	=	2700/61
K	=	4	LC	B	(2)	=	-327096/3355
K	=	4	ALPHA	(2)	=	1080/3943	
K	=	4	BETA	(2)	=	64152/216865	
K	=	4	CAP	A	(3)	=	898425/59049
K	=	4	CAP	B	(3)	=	217800/19683
K	=	4	LC	A	(3)	=	-5204/61
K	=	4	LC	B	(3)	=	-210705/3355
K	=	4	ALPHA	(3)	=	568/3943	
K	=	4	BETA	(3)	=	14190/216865	
K	=	4	CAP	A	(4)	=	208000/59049
K	=	4	CAP	B	(4)	=	17600/19683
K	=	4	LC	A	(4)	=	-1231/61
K	=	4	LC	B	(4)	=	-17220/3355
K	=	4	ALPHA	(4)	=	47/3943	
K	=	4	BETA	(4)	=	570/216865	
K	=	4	LC	B	=	6561/3355	

K = 4	BETA =	118098/216865
K = 4	BETA (0) =	20130/216865
K = 5	CAP A (1) =	-9486400/531441
K = 5	CAP B (1) =	1185800/177147
K = 5	LC A (1) =	454750/4503
K = 5	LC B (1) =	-44017050/1386924
K = 5	ALPHA (1) =	230875/479327
K = 5	BETA (1) =	46719750/73816358
K = 5	CAP A (2) =	-24285184/531441
K = 5	CAP B (2) =	7589120/177147
K = 5	LC A (2) =	1181300/4503
K = 5	LC B (2) =	-328685280/1386924
K = 5	ALPHA (2) =	97000/479327
K = 5	BETA (2) =	32062800/73816358
K = 5	CAP A (3) =	12006225/531441
K = 5	CAP B (3) =	10672200/177147
K = 5	LC A (3) =	-558150/4503
K = 5	LC B (3) =	-472227525/1386924
K = 5	ALPHA (3) =	117500/479327
K = 5	BETA (3) =	13369125/73816358
K = 5	CAP A (4) =	20070400/531441
K = 5	CAP B (4) =	3449600/177147
K = 5	LC A (4) =	-965725/4503
K = 5	LC B (4) =	-153728400/1386924
K = 5	ALPHA (4) =	32375/479327
K = 5	BETA (4) =	1879500/73816358
K = 5	CAP A (5) =	2226400/531441
K = 5	CAP B (5) =	169400/177147
K = 5	LC A (5) =	-107672/4503
K = 5	LC B (5) =	-7573830/1386924
K = 5	ALPHA (5) =	1577/479327
K = 5	BETA (5) =	49170/73816358
K = 5	LC B =	2657205/1386924
K = 5	BETA =	39858075/73816358
K = 5	BETA (0) =	6934620/73816358
K = 6	CAP A (1) =	-630561008/23914845
K = 6	CAP B (1) =	13707848/1594323
K = 6	LC A (1) =	21459012/142985
K = 6	LC B (1) =	-1599082254/37433473
K = 6	ALPHA (1) =	1485084/3044563

K	=	6	BETA (1)	=	2514772260/3985332967
K	=	6	CAP A (2)	=	-3221344280/23914845
K	=	6	CAP B (2)	=	137078480/1594323
K	=	6	LC A (2)	=	110802000/142985
K	=	6	LC B (2)	=	-18173488410/37433473
K	=	6	ALPHA (2)	=	694875/3044563
K	=	6	BETA (2)	=	1660531950/3985332967
K	=	6	CAP A (3)	=	-1499295875/23914845
K	=	6	CAP B (3)	=	342696200/1594323
K	=	6	LC A (3)	=	52489000/142985
K	=	6	LC B (3)	=	-46096009575/37433473
K	=	6	ALPHA (3)	=	774000/3044563
K	=	6	BETA (3)	=	559924750/3985332967
K	=	6	CAP A (4)	=	3511928000/23914845
K	=	6	CAP B (4)	=	249233600/1594323
K	=	6	LC A (4)	=	-120483375/142985
K	=	6	LC B (4)	=	-33650498700/37433473
K	=	6	ALPHA (4)	=	121875/3044563
K	=	6	BETA (4)	=	-9906750/3985332967
K	=	6	CAP A (5)	=	1748450000/23914845
K	=	6	CAP B (5)	=	48956600/1594323
K	=	6	LC A (5)	=	-60172140/142985
K	=	6	LC B (5)	=	-6618347010/37433473
K	=	6	ALPHA (5)	=	-28812/3044563
K	=	6	BETA (5)	=	-18412020/3985332967
K	=	6	CAP A (6)	=	114738008/23914845
K	=	6	CAP B (6)	=	1612688/1594323
K	=	6	LC A (6)	=	-3951512/142985
K	=	6	LC B (6)	=	-218118054/37433473
K	=	6	ALPHA (6)	=	-2459/3044563
K	=	6	BETA (6)	=	-675290/3985332967
K	=	6	LC B	=	71744535/37433473
K	=	6	BETA	=	2152336050/3985332967
K	=	6	BETA (0)	=	374334730/3985332967
M	=	2	P (0)	=	2837/1440
M	=	2	P (1)	=	-2543/720
M	=	2	P (2)	=	17/5
M	=	2	P (3)	=	-1201/720
M	=	2	P (4)	=	95/288
M	=	2	C (0)	=	95/288

M	=	2	G	(1)	=	77/240
M	=	2	C	(2)	=	-7/30
M	=	2	C	(3)	=	73/720
M	=	2	C	(4)	=	-3/160
M	=	2	M	(0)	=	11/1440
M	=	2	M	(1)	=	-41/720
M	=	2	M	(2)	=	1/2
M	=	2	M	(3)	=	41/720
M	=	2	M	(4)	=	-11/1440
M	=	3	P	(0)	=	11603/4480
M	=	3	P	(1)	=	-104861/15120
M	=	3	P	(2)	=	1344989/120960
M	=	3	P	(3)	=	-20617/1890
M	=	3	P	(4)	=	156551/24192
M	=	3	P	(5)	=	-32371/15120
M	=	3	P	(6)	=	5257/17280
M	=	3	C	(0)	=	5257/17280
M	=	3	C	(1)	=	6961/15120
M	=	3	C	(2)	=	-66109/120960
M	=	3	C	(3)	=	33/70
M	=	3	C	(4)	=	-31523/120960
M	=	3	C	(5)	=	1247/15120
M	=	3	C	(6)	=	-275/24192
M	=	3	M	(0)	=	-191/120960
M	=	3	M	(1)	=	211/15120
M	=	3	M	(2)	=	-7843/120960
M	=	3	M	(3)	=	1/2
M	=	3	M	(4)	=	7843/120960
M	=	3	M	(5)	=	-211/15120
M	=	3	M	(6)	=	191/120960
M	=	4	P	(0)	=	3288521/1036800
M	=	4	P	(1)	=	-40987771/3628800
M	=	4	P	(2)	=	10219841/403200
M	=	4	P	(3)	=	-135352319/3628800
M	=	4	P	(4)	=	167287/4536
M	=	4	P	(5)	=	-9839609/403200
M	=	4	P	(6)	=	5393233/518400
M	=	4	P	(7)	=	-9401029/3628800
M	=	4	P	(8)	=	25713/89600
M	=	4	C	(0)	=	25713/89600

M	=	4	C	(1)	=	427487/725760
M	=	4	C	(2)	=	-3498217/3628800
M	=	4	C	(3)	=	500327/403200
M	=	4	C	(4)	=	-6467/5670
M	=	4	C	(5)	=	2616161/3628800
M	=	4	C	(6)	=	-24019/80640
M	=	4	C	(7)	=	263077/3628800
M	=	4	C	(8)	=	-8183/1036800
M	=	4	M	(0)	=	2497/7257600
M	=	4	M	(1)	=	-1469/403200
M	=	4	M	(2)	=	68119/3628800
M	=	4	M	(3)	=	-252769/3628800
M	=	4	M	(4)	=	$\frac{1}{2}$
M	=	4	M	(5)	=	252769/3628800
M	=	4	M	(6)	=	-68119/3628800
M	=	4	M	(7)	=	1469/403200
M	=	4	M	(8)	=	-2497/7257600
M	=	5	P	(0)	=	3569763199/958003200
M	=	5	P	(1)	=	-3966011741/239500800
M	=	5	P	(2)	=	1695154823/35481600
M	=	5	P	(3)	=	-1247363563/13305600
M	=	5	P	(4)	=	4144305961/31933440
M	=	5	P	(5)	=	-495967/3850
M	=	5	P	(6)	=	2087883637/22809600
M	=	5	P	(7)	=	-86656259/1900800
M	=	5	P	(8)	=	76795519/5068800
M	=	5	P	(9)	=	-144794759/47900160
M	=	5	P	(10)	=	4777223/17418240
M	=	5	C	(0)	=	4777223/17418240
M	=	5	C	(1)	=	8089801/11404800
M	=	5	C	(2)	=	-67283209/45619200
M	=	5	C	(3)	=	14380247/5702400
M	=	5	C	(4)	=	-517263181/159667200
M	=	5	C	(5)	=	76561/24948
M	=	5	C	(6)	=	-337204019/159667200
M	=	5	C	(7)	=	41021471/39916800
M	=	5	C	(8)	=	-107151937/319334400
M	=	5	C	(9)	=	15813379/239500800
M	=	5	C	(10)	=	-4671/788480
M	=	5	M	(0)	=	-14797/191600640

M	=	5	M	(1)	=	M	(1)	=	230371/239500800
M	=	5	M	(2)	=	M	(2)	=	-203257/35481600
M	=	5	M	(3)	=	M	(3)	=	299093/13305600
M	=	5	M	(4)	=	M	(4)	=	-11639731/159667200
M	=	5	M	(5)	=	M	(5)	=	1/2
M	=	5	M	(6)	=	M	(6)	=	11639731/159667200
M	=	5	M	(7)	=	M	(7)	=	-299093/13305600
M	=	5	M	(8)	=	M	(8)	=	203257/35481600
M	=	5	M	(9)	=	M	(9)	=	-230371/239500800
M	=	5	M	(10)	=	M	(10)	=	14797/191600640
P	=	6	P	(0)	=	P	(0)	=	733526173/172204032
P	=	6	P	(1)	=	P	(1)	=	-59344946587373/2615348736000
P	=	6	P	(2)	=	P	(2)	=	104639289835229/1307674368000
P	=	6	P	(3)	=	P	(3)	=	-102675619234099/523069747200
P	=	6	P	(4)	=	P	(4)	=	121844891963321/348713164800
P	=	6	P	(5)	=	P	(5)	=	-40318232897599/87178291200
P	=	6	P	(6)	=	P	(6)	=	31975145483/69498000
P	=	6	P	(7)	=	P	(7)	=	-149831214658501/435891456000
P	=	6	P	(8)	=	P	(8)	=	66393001798471/348713164800
P	=	6	P	(9)	=	P	(9)	=	-13247042672623/174356582400
P	=	6	P	(10)	=	P	(10)	=	491703913717/23775897600
P	=	6	P	(11)	=	P	(11)	=	-900005832083/2615348736000
P	=	6	P	(12)	=	P	(12)	=	106364763817/402361344000
C	=	6	C	(0)	=	C	(0)	=	106364763817/402361344000
C	=	6	C	(1)	=	C	(1)	=	307515172843/373621248000
C	=	6	C	(2)	=	C	(2)	=	-2709005666077/1307674368000
C	=	6	C	(3)	=	C	(3)	=	2309296746931/523069747200
C	=	6	C	(4)	=	C	(4)	=	-507942835493/69742632960
C	=	6	C	(5)	=	C	(5)	=	4007043002299/435891456000
C	=	6	C	(6)	=	C	(6)	=	-2215533/250250
C	=	6	C	(7)	=	C	(7)	=	2816016533573/435891456000
C	=	6	C	(8)	=	C	(8)	=	-175102023617/49816166400
C	=	6	C	(9)	=	C	(9)	=	144690945961/104613949440
C	=	6	C	(10)	=	C	(10)	=	-486772076771/1307674368000
C	=	6	C	(11)	=	C	(11)	=	160495253651/2615348736000
C	=	6	C	(12)	=	C	(12)	=	-2224234463/475517952000
M	=	6	M	(0)	=	M	(0)	=	92427157/5230697472000
M	=	6	M	(1)	=	M	(1)	=	-132822967/523069747200
M	=	6	M	(2)	=	M	(2)	=	2274524387/1307674368000
M	=	6	M	(3)	=	M	(3)	=	-4013113421/523069747200

M	=	6	M	(4)	=
M	=	6	M	(5)	=
M	=	6	M	(6)	=
M	=	6	M	(7)	=
M	=	6	M	(8)	=
M	=	6	M	(9)	=
M	=	6	M	(10)	=
M	=	6	M	(11)	=
M	=	6	M	(12)	=
M	=	7	P	(0)	=
M	=	7	P	(1)	=
M	=	7	P	(2)	=
M	=	7	P	(3)	=
M	=	7	P	(4)	=
M	=	7	P	(5)	=
M	=	7	P	(6)	=
M	=	7	P	(7)	=
M	=	7	P	(8)	=
M	=	7	P	(9)	=
M	=	7	P	(10)	=
M	=	7	P	(11)	=
M	=	7	P	(12)	=
M	=	7	P	(13)	=
M	=	7	P	(14)	=
M	=	7	C	(0)	=
M	=	7	C	(1)	=
M	=	7	C	(2)	=
M	=	7	C	(3)	=
M	=	7	C	(4)	=
M	=	7	C	(5)	=
M	=	7	C	(6)	=
M	=	7	C	(7)	=
M	=	7	C	(8)	=
M	=	7	C	(9)	=
M	=	7	C	(10)	=
M	=	7	C	(11)	=
M	=	7	C	(12)	=
M	=	7	C	(13)	=
M	=	7	C	(14)	=
M	=	7	M	(0)	=

8855328071/348713164800
-32793164357/435891456000
32793164357/435891456000
-8855328071/348713164800
4013113421/523069747200
-2274524387/1307674368000
132822967/523069747200
-92427157/5230697472000
299786756763073/62768369664000
-116361307155361/3923023104000
2586771998343187/20922789888000
-356985279148297/980755776000
1988442368270749/2510734786560
-571195366208749/435891456000
5010047970421097/2988969984000
-2132356395131/1277025750
9030884747790859/6974263296000
-121630328435299/156920924160
2006565473520353/5706215424000
-3478073249303/29719872000
130221619246627/4828336128000
-2156804881129/560431872000
25221445/98402304
25221445/98402304
3654051145153/3923023104000
-172527345401401/62768369664000
2292797894083/326918592000
-886761467394133/62768369664000
3496017827389/156920924160
-192338437893109/6974263296000
349581097/13030875
-427489980816929/20922789888000
5256082896499/435891456000
-13579171932259/2510734786560
1748809541047/980755776000
-8530634387437/20922789888000
22671757011/3923023104000
-2639651053/689762304000
36740617/8966909952000

M	= 7	M	(1)	=	87402869/1307674368000
M	= 7	M	(2)	=	-1306229471/2510734786560
M	= 7	M	(3)	=	2541742327/980755776000
M	= 7	M	(4)	=	-197301894457/20922789888000
M	= 7	M	(5)	=	108816780203/3923023104000
M	= 7	M	(6)	=	-1610849246753/20922789888000
M	= 7	M	(7)	=	1/2
M	= 7	M	(8)	=	1610849246753/20922789888000
M	= 7	M	(9)	=	-108816780203/3923023104000
M	= 7	M	(10)	=	197301894457/20922789888000
M	= 7	M	(11)	=	-2541742327/980755776000
M	= 7	M	(12)	=	1306229471/2510734786560
M	= 7	M	(13)	=	-87402869/1307674368000
M	= 7	M	(14)	=	36740617/8966909952000
M	= 8	P	(0)	=	48278377805453833/9146248151040000
M	= 8	P	(1)	=	-171249214157564497/4573124075520000
M	= 8	P	(2)	=	469380177761711/2605768704000
M	= 8	P	(3)	=	-3961751682437057363/6402373705728000
M	= 8	P	(4)	=	10188305820220195813/6402373705728000
M	= 8	P	(5)	=	-3746390185754199257/1185624760320000
M	= 8	P	(6)	=	22592520393618350801/4573124075520000
M	= 8	P	(7)	=	-39387573858057739199/6402373705728000
M	= 8	P	(8)	=	11690920326343/1905904000
M	= 8	P	(9)	=	-31344919029592580161/6402373705728000
M	= 8	P	(10)	=	9049517901190374779/2910169866240000
M	= 8	P	(11)	=	-1840516046810912551/1185624760320000
M	= 8	P	(12)	=	293655970246750919/492490285056000
M	= 8	P	(13)	=	-14149115258073569/83147710464000
M	= 8	P	(14)	=	8062298103159499/237124952064000
M	= 8	P	(15)	=	-135934383865740233/32011868528640000
M	= 8	P	(16)	=	85455477715379/342372925440000
M	= 8	C	(0)	=	85455477715379/342372925440000
M	= 8	C	(1)	=	736507566455411/711374856192000
M	= 8	C	(2)	=	-2490947654982047/711374856192000
M	= 8	C	(3)	=	66615242131764563/6402373705728000
M	= 8	C	(4)	=	-17607799026266621/711374856192000
M	= 8	C	(5)	=	166541079499158667/3556874280960000
M	= 8	C	(6)	=	-453443248829255563/6402373705728000
M	= 8	C	(7)	=	20417981803080493/237124952064000
M	= 8	C	(8)	=	-610091660201/7236479250

M	= 8	C (9)	=	424709866723701313/6402373705728000
M	= 8	C (10)	=	-148153326227812417/3556874280960000
M	= 8	C (11)	=	1332077054297011/64670441472000
M	= 8	C (12)	=	-50254775657217563/6402373705728000
M	= 8	C (13)	=	1582902445233797/711374856192000
M	= 8	C (14)	=	-28586063059651/64670441472000
M	= 8	C (15)	=	1758389297773001/32011868528640000
M	= 8	C (16)	=	-50188465/15613165568
M	= 8	M (0)	=	61430943169/64023737057280000
M	= 8	M (1)	=	-80168657839/4573124075520000
M	= 8	M (2)	=	660473357/4311362764800
M	= 8	M (3)	=	-1096193632393/1280474741145600
M	= 8	M (4)	=	4436541947807/1280474741145600
M	= 8	M (5)	=	-13043845962023/1185624760320000
M	= 8	M (6)	=	949437300568649/32011868528640000
M	= 8	M (7)	=	-14334414125131/182924963020800
M	= 8	M (8)	=	1/2
M	= 8	M (9)	=	14334414125131/182924963020800
M	= 8	M (10)	=	-949437300568649/32011868528640000
M	= 8	M (11)	=	13043845962023/1185624760320000
M	= 8	M (12)	=	-4436541947807/1280474741145600
M	= 8	M (13)	=	1096193632393/1280474741145600
M	= 8	M (14)	=	-660473357/4311362764800
M	= 8	M (15)	=	80168657839/4573124075520000
M	= 8	M (16)	=	-61430943169/64023737057280000
	(4,4)	B(1,0)	=	1/100
	(4,4)	A(1)	=	1/100
	(4,4)	B(2,0)	=	-4278/245
	(4,4)	B(2,1)	=	4425/245
	(4,4)	A(2)	=	3/5
	(4,4)	B(3,0)	=	524746/8791
	(4,4)	B(3,1)	=	-532125/8791
	(4,4)	B(3,2)	=	16170/8791
	(4,4)	A(3)	=	1/1
	(4,4)	B(4,0)	=	-179124/70092
	(4,4)	B(4,1)	=	20000/70092
	(4,4)	B(4,2)	=	40425/70092
	(4,4)	B(4,3)	=	8791/70092
	(5,5)	B(1,0)	=	1/1000
	(5,5)	A(1)	=	1/1000

(5,5)	B(2,0)	-447/10
(5,5)	B(2,1)	450/10
(5,5)	A(2)	3/10
(5,5)	B(3,0)	2684721/9568
(5,5)	B(3,1)	-2695500/9568
(5,5)	B(3,2)	17955/9568
(5,5)	A(3)	3/4
(5,5)	B(4,0)	-30872049/24219
(5,5)	B(4,1)	31009500/24219
(5,5)	B(4,2)	-146720/24219
(5,5)	B(4,3)	33488/24219
(5,5)	A(4)	1/1
(5,5)	B(5,0)	105/1134
(5,5)	B(5,1)	0/1134
(5,5)	B(5,2)	500/1134
(5,5)	B(5,3)	448/1134
(5,5)	B(5,4)	81/1134
(5,5)	B(1,0)	1/300
(6,6)	A(1)	1/300
(6,6)	B(2,0)	-29/5
(6,6)	B(2,1)	30/5
(6,6)	A(2)	1/5
(6,6)	B(3,0)	323/5
(6,6)	B(3,1)	-330/5
(6,6)	B(3,2)	10/5
(6,6)	A(3)	3/5
(6,6)	B(4,0)	-510104/810
(6,6)	B(4,1)	521640/810
(6,6)	B(4,2)	-12705/810
(6,6)	B(4,3)	1925/810
(6,6)	A(4)	14/15
(6,6)	B(5,0)	-417923/77
(6,6)	B(5,1)	427350/77
(6,6)	B(5,2)	-10605/77
(6,6)	B(5,3)	1309/77
(6,6)	B(5,4)	-54/77
(6,6)	A(5)	1/1
(6,6)	B(6,0)	198/3696
(6,6)	B(6,1)	0/3696
(6,6)	B(6,2)	1225/3696

(6,6)	B(6,3)	1540/3696
(6,6)	B(6,4)	810/3696
(6,6)	B(6,5)	-77/3696
(7,7)	B(1,0)	1/192
(7,7)	A(1)	1/192
(7,7)	B(2,0)	-15/6
(7,7)	B(2,1)	16/6
(7,7)	A(2)	1/6
(7,7)	B(3,0)	4867/186
(7,7)	B(3,1)	-5072/186
(7,7)	B(3,2)	298/186
(7,7)	A(3)	1/2
(7,7)	B(4,0)	-19995/31
(7,7)	B(4,1)	20896/31
(7,7)	B(4,2)	-1025/31
(7,7)	B(4,3)	155/31
(7,7)	A(4)	1/1
(7,7)	B(5,0)	-469805/5022
(7,7)	B(5,1)	490960/5022
(7,7)	B(5,2)	-22736/5022
(7,7)	B(5,3)	5580/5022
(7,7)	B(5,4)	186/5022
(7,7)	A(5)	5/6
(7,7)	B(6,0)	914314/2604
(7,7)	B(6,1)	-955136/2604
(7,7)	B(6,2)	47983/2604
(7,7)	B(6,3)	-6510/2604
(7,7)	B(6,4)	-558/2604
(7,7)	B(6,5)	2511/2604
(7,7)	A(6)	1/1
(7,7)	B(7,0)	14/300
(7,7)	B(7,1)	0/300
(7,7)	B(7,2)	81/300
(7,7)	B(7,3)	110/300
(7,7)	B(7,4)	0/300
(7,7)	B(7,5)	81/300
(7,7)	B(7,6)	14/300
(7,9)	B(1,0)	2/9
(7,9)	A(1)	2/9
(7,9)	B(2,0)	1/12

(7,9)	B(2,1)	3/12
(7,9)	A(2)	1/3
(7,9)	B(3,0)	1/8
(7,9)	B(3,1)	0/8
(7,9)	B(3,2)	3/8
(7,9)	A(3)	1/2
(7,9)	B(4,0)	23/216
(7,9)	B(4,1)	0/216
(7,9)	B(4,2)	21/216
(7,9)	B(4,3)	-8/216
(7,9)	A(4)	1/6
(7,9)	B(5,0)	-4136/729
(7,9)	B(5,1)	0/729
(7,9)	B(5,2)	-13584/729
(7,9)	B(5,3)	5264/729
(7,9)	B(5,4)	13104/729
(7,9)	A(5)	8/9
(7,9)	B(6,0)	105131/151632
(7,9)	B(6,1)	0/151632
(7,9)	B(6,2)	302016/151632
(7,9)	B(6,3)	-107744/151632
(7,9)	B(6,4)	-284256/151632
(7,9)	B(6,5)	1701/151632
(7,9)	A(6)	1/9
(7,9)	B(7,0)	-775229/1375920
(7,9)	B(7,1)	0/1375920
(7,9)	B(7,2)	-2770950/1375920
(7,9)	B(7,3)	1735136/1375920
(7,9)	B(7,4)	2547216/1375920
(7,9)	B(7,5)	81891/1375920
(7,9)	B(7,6)	328536/1375920
(7,9)	A(7)	5/6
(7,9)	B(8,0)	23569/251888
(7,9)	B(8,1)	0/251888
(7,9)	B(8,2)	-122304/251888
(7,9)	B(8,3)	-20384/251888
(7,9)	B(8,4)	695520/251888
(7,9)	B(8,5)	-99873/251888
(7,9)	B(8,6)	-466560/251888
(7,9)	B(8,7)	241920/251888

(7,9)	A(8)	1/1	110201/2140320
(7,9)	B(9,0)		0/2140320
(7,9)	B(9,1)		0/2140320
(7,9)	B(9,2)		767936/2140320
(7,9)	B(9,3)		635040/2140320
(7,9)	B(9,4)		-59049/2140320
(7,9)	B(9,5)		-59049/2140320
(7,9)	B(9,6)		635040/2140320
(7,9)	B(9,7)		110201/2140320
(7,9)	B(9,8)	4/27	
(8,10)	B(1,0)	4/27	
(8,10)	A(1)	1/18	
(8,10)	B(2,0)	3/18	
(8,10)	B(2,1)	2/9	
(8,10)	A(2)	1/12	
(8,10)	B(3,0)	0/12	
(8,10)	B(3,1)	3/12	
(8,10)	B(3,2)	1/3	
(8,10)	A(3)	1/8	
(8,10)	B(4,0)	0/8	
(8,10)	B(4,1)	0/8	
(8,10)	B(4,2)	3/8	
(8,10)	B(4,3)	1/2	
(8,10)	A(4)	13/54	
(8,10)	B(5,0)	0/54	
(8,10)	B(5,1)	-27/54	
(8,10)	B(5,2)	42/54	
(8,10)	B(5,3)	8/54	
(8,10)	B(5,4)	2/3	
(8,10)	A(5)	389/4320	
(8,10)	B(6,0)	0/4320	
(8,10)	B(6,1)	-54/4320	
(8,10)	B(6,2)	966/4320	
(8,10)	B(6,3)	-824/4320	
(8,10)	B(6,4)	243/4320	
(8,10)	B(6,5)	1/6	
(8,10)	A(6)	-231/20	
(8,10)	B(7,0)	0/20	
(8,10)	B(7,1)	81/20	
(8,10)	B(7,2)		

(8,10)	B(7,3)	-1164/20
(8,10)	B(7,4)	656/20
(8,10)	B(7,5)	-122/20
(8,10)	B(7,6)	800/20
(8,10)	A(7)	1/1
(8,10)	B(8,0)	-127/288
(8,10)	B(8,1)	0/288
(8,10)	B(8,2)	18/288
(8,10)	B(8,3)	-678/288
(8,10)	B(8,4)	456/288
(8,10)	B(8,5)	-9/288
(8,10)	B(8,6)	576/288
(8,10)	B(8,7)	4/288
(8,10)	A(8)	5/6
(8,10)	B(9,0)	1481/820
(8,10)	B(9,1)	0/820
(8,10)	B(9,2)	-81/820
(8,10)	B(9,3)	7104/820
(8,10)	B(9,4)	-3376/820
(8,10)	B(9,5)	72/820
(8,10)	B(9,6)	-5040/820
(8,10)	B(9,7)	-60/820
(8,10)	B(9,8)	720/820
(8,10)	A(9)	1/1
(8,10)	B(10,0)	41/840
(8,10)	B(10,1)	0/840
(8,10)	B(10,2)	0/840
(8,10)	B(10,3)	27/840
(8,10)	B(10,4)	272/840
(8,10)	B(10,5)	27/840
(8,10)	B(10,6)	216/840
(8,10)	B(10,7)	0/840
(8,10)	B(10,8)	216/840
(8,10)	B(10,9)	41/840
(8,12)	B(1, 0)	1/9
(8,12)	A(1)	1/9
(8,12)	B(2, 0)	1/24
(8,12)	B(2, 1)	3/24
(8,12)	A(2)	1/6
(8,12)	B(3, 0)	1/16

(8,12)	B(3, 1)	0/16
(8,12)	B(3, 2)	3/16
(8,12)	A(3)	1/4
(8,12)	B(4, 0)	29/500
(8,12)	B(4, 1)	0/500
(8,12)	B(4, 2)	33/500
(8,12)	B(4, 3)	-12/500
(8,12)	A(4)	1/10
(8,12)	B(5, 0)	33/972
(8,12)	B(5, 1)	0/972
(8,12)	B(5, 2)	0/972
(8,12)	B(5, 3)	4/972
(8,12)	B(5, 4)	125/972
(8,12)	A(5)	1/6
(8,12)	B(6, 0)	-21/36
(8,12)	B(6, 1)	0/36
(8,12)	B(6, 2)	0/36
(8,12)	B(6, 3)	76/36
(8,12)	B(6, 4)	125/36
(8,12)	B(6, 5)	-162/36
(8,12)	A(6)	1/2
(8,12)	B(7, 0)	-30/243
(8,12)	B(7, 1)	0/243
(8,12)	B(7, 2)	0/243
(8,12)	B(7, 3)	-32/243
(8,12)	B(7, 4)	125/243
(8,12)	B(7, 5)	0/243
(8,12)	B(7, 6)	99/243
(8,12)	A(7)	2/3
(8,12)	B(8, 0)	1175/324
(8,12)	B(8, 1)	0/324
(8,12)	B(8, 2)	0/324
(8,12)	B(8, 3)	-3456/324
(8,12)	B(8, 4)	-6250/324
(8,12)	B(8, 5)	8424/324
(8,12)	B(8, 6)	242/324
(8,12)	B(8, 7)	-27/324
(8,12)	A(8)	1/3
(8,12)	B(9, 0)	293/324
(8,12)	B(9, 1)	0/324

(8,12)	B(9, 2)	0/324
(8,12)	B(9, 3)	-852/324
(8,12)	B(9, 4)	-1375/324
(8,12)	B(9, 5)	1836/324
(8,12)	B(9, 6)	-118/324
(8,12)	B(9, 7)	162/324
(8,12)	B(9, 8)	324/324
(8,12)	A(9)	5/6
(8,12)	B(10, 0)	1303/1620
(8,12)	B(10, 1)	0/1620
(8,12)	B(10, 2)	0/1620
(8,12)	B(10, 3)	-4260/1620
(8,12)	B(10, 4)	-6875/1620
(8,12)	B(10, 5)	990/1620
(8,12)	B(10, 6)	1030/1620
(8,12)	B(10, 7)	0/1620
(8,12)	B(10, 8)	0/1620
(8,12)	B(10, 9)	162/1620
(8,12)	A(10)	5/6
(8,12)	B(11, 0)	-8595/4428
(8,12)	B(11, 1)	0/4428
(8,12)	B(11, 2)	0/4428
(8,12)	B(11, 3)	30720/4428
(8,12)	B(11, 4)	48750/4428
(8,12)	B(11, 5)	-66096/4428
(8,12)	B(11, 6)	378/4428
(8,12)	B(11, 7)	-729/4428
(8,12)	B(11, 8)	-1944/4428
(8,12)	B(11, 9)	-1296/4428
(8,12)	B(11,10)	3240/4428
(8,12)	A(11)	1/1
(8,12)	B(12, 0)	41/840
(8,12)	B(12, 1)	0/840
(8,12)	B(12, 2)	0/840
(8,12)	B(12, 3)	0/840
(8,12)	B(12, 4)	0/840
(8,12)	B(12, 5)	216/840
(8,12)	B(12, 6)	272/840
(8,12)	B(12, 7)	27/840
(8,12)	B(12, 8)	27/840

$(8, 12)$ $B(12, 9)$
 $(8, 12)$ $B(12, 10)$
 $(8, 12)$ $B(12, 11)$

$36/840$
 $180/840$
 $41/840$

V. RESULTS AND CONCLUSIONS

A. Applications

Three types of problems were used to exercise this integration procedure. The first type is the Arenstorf orbits of the restricted three body problem. The second is the system of linear differential equations associated with Fourier transforms. The third type is the system of linear equations obtained from a discretization of the partial differential equation for the vibrating string. The first of these is characterized by the necessity of frequent step-size change. The other two types are characterized by having a large number (20 to 100) of coupled equations. The accuracy range studied was 10^{-3} to 10^{-18} .

B. Results

The executive routine performed quite satisfactorily. Learning took place as was desired, the procedure adapting readily to each problem type.

All integration methods performed well. For a given problem type one particular method and order usually dominated, but which one proved superior depended on the problem type. For example, the Runge-Kutta-Shanks formulas were usually faster for problems where frequent step-size change was required and large error tolerances were acceptable, but the multi-step methods performed better where small error tolerances were required or where long runs of uniform step size were appropriate. Example histories for four representative problem types are given in Tables III to VI.

Other results concerning effective use of step-size and error control, and

TABLE III

EXAMPLE HISTORY OF COMPETITIONS BETWEEN METHODS

vs	Adams		Butcher		Cowell		Runge-Kutta	
	W	L	W	L	W	L	W	L
Adams	*	*	0	0	10573	-5036	0	0
Butcher	0	0	*	*	10017	-6490	0	0
Cowell	4388	-14241	5908	-15101	*	*	0	-10834
Shanks	0	0	0	0	7138	0	*	*

The entries in the table represent net times (in 1/60 sec) spent winning or losing (+ for win, - for loss) against each opponent. The problem type represented here is: greater than 6 equations, time for a function evaluation short, and an error tolerance between 10^{-6} and 10^{-3} . 202 single precision competitions are represented here.

TABLE IV

EXAMPLE HISTORY OF COMPETITIONS BETWEEN METHODS

vs	Adams		Butcher		Cowell		Runge-Kutta	
	W	L	W	L	W	L	W	L
Adams	*	*	0	0	0	0	1047	-157
Butcher	0	0	*	*	71	0	271	0
Cowell	0	0	0	-560	*	*	1005	-46
Shanks	142	-1478	0	-1490	43	-1576	*	*

The entries in the table represent net times (in 1/60 sec) spent winning and losing (+ for win, - for loss) against each opponent. The problem type represented here is: 6 or less equations, time for a function evaluation long, and an error tolerance $< 10^{-6}$. 47 single precision competitions are represented here.

TABLE V

EXAMPLE HISTORY OF COMPETITIONS BETWEEN METHODS

vs	Adams		Butcher		Cowell		Runge-Kutta	
	W	L	W	L	W	L	W	L
Adams	*	*	0	0	2634	0	0	0
Butcher	0	0	*	*	2531	-1294	0	0
Cowell	0	-3469	1100	-3595	*	*	0	-3229
Shanks	0	0	0	0	1811	0	*	*

The entries in the table represent net times (in 1/60 sec) spent winning and losing (+ for win, - for loss) against each opponent. The problem type represented here is: 6 or less equations, time for a function evaluation large, and an error tolerance $\sim 10^{-8}$. 30 double precision competitions are represented here.

TABLE VI

EXAMPLE HISTORY OF COMPETITIONS BETWEEN METHODS

vs	Adams		Butcher		Cowell		Runge-Kutta	
	W	L	W	L	W	L	W	L
Adams	*	*	2378	0	0	0	0	0
Butcher	0	-3605	*	*	738	-2601	0	-5218
Cowell	0	0	1889	-974	*	*	0	0
Shanks	0	0	1652	0	0	0	*	*

The entries in the table represent net times (in 1/60 sec) spent winning or losing (+ for win, - for loss) against each opponent. The problem type represented here is: 6 or less equations, time for a function evaluation long, and an error tolerance between 10^{-8} and 10^{-14} . 19 double precision competitions are represented here.

tests of accuracies and efficiency as a function of order for the individual methods are discussed in Chapter II in the sections describing that method.

C. Conclusions

The results justify the conclusion that the present programs would be suitable and effective as a general library programs for integrating systems of differential equations. It was evident that no particular method or order is exceptionally superior to all the others. Depending on the accuracy and the problem, different methods and orders are best. The executive routine does an effective job of finding a good method and order for each individual problem type.

D. Recommendations for Further Study

Improvements or modifications to the present program and additional tasks that would be fruitful are as follows:

The classification mechanism of the present program could be sharpened and expanded. For example the classification could take into account the name of the person or organizational element submitting the problem and keep history files based on this additional information.

Hybridization of the single and double-precision procedures into one procedure that allows the program to make the decision about when to use single and when to use double. (This may be difficult or awkward with present computer languages.)

Other integration methods [19] could be added to, or replace some of, the methods now in use and tests made to evaluate their effectiveness.

This or a similar program could be converted to other machines (such as the Univac 1108). The relative efficiency of the methods might also be machine dependent, and this should be studied. (Since the procedure is in Algol, its conversion to other hardware should be relatively straightforward.)

It would be of interest to make this program available to an on-line system (remote terminal or time sharing system) as an intrinsic procedure for integrating differential equations and see how effective or useful it would be in this role.

Finally, it would be of interest to compare the use of this procedure with other methods of solving boundary value problems and integral equations.

VI. BIBLIOGRAPHY

References on the Adams Method

1. Krogh, Fred T., J. ACM, 13, (1966) 374-385.
2. Bashforth, F., and Adams, J. C., Theories of Capillary Action, Cambridge University Press, 1883.
3. Dahlquist, Germund, Math. Scan., 4, (1956) 33.
4. Dahlquist, Germund, Stockholm Tekniska Hogskolan, No. 130, (1959), 1-87.
5. Henrici, Peter, Discrete Variable Methods in Ordinary Differential Equations, John Wiley and Sons, New York, 1962, 191-199, 200, 203-204, 224, 241, 258, 272-273, 274.
6. Newberry, A. C. R., Math. Comp. 17, 84, (1963) 452-455.

References on Cowell (Constant Nth Order Difference) Method

7. Cowell, P. H., and Crommelin, A. C. D., "Investigation of the Motion of Halley's Comet from 1759 to 1910." Appendix to Greenwich Observations for 1909, Edinburgh, 1910, 84.
8. Herrick, S., "Step-by-Step Integration of $\dot{x} = f(x, y, z, t)$ without a Corrector," Mathematical Tables and Other Aids to Computation, No. 34, April 1951, 61-67.
9. Durham, H. L., Jr., et al, "Study of Methods for Numerical Solution of Ordinary Differential Equations," Final Report, Contract NAS8-11129, Project A-740, Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Georgia, 1964.
10. Currie, J. C., et al, "Study of Satellite Orbit Computation Methods and An Error Analysis of the Mathematical Procedures," Technical Report No. 3, Prepared for Department of the Air Force Contract No. AF29(600)-1756, Project No. 1770, Headquarters Air Force Missile Development Center, Air Research and Development Command, Holloman Air Force Base, New Mexico, August 1959.
11. Francis, O. B., Jr., et al, "Study of the Methods for the Numerical Solution of Ordinary Differential Equations," Final Report, Project A-831, Contract NAS8-20014, Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Georgia, 1966.

References on the Stetter-Gragg-Butcher Method

12. Gear, C. W., "Hybrid Methods for Initial Value Problems in Ordinary Differential Equations," J.SIAM (B) 2 (1965) p. 69.
13. Butcher, J. C., "A Modified Multistep Method for the Numerical Integration of Ordinary Differential Equations," J. ACM, 12 (1965) 124-135.
14. Gragg, W. B. and Stetter, H. J., "Generalized Multistep Predictor-Corrector Methods," J. ACM, 11 (1964) 188-209.

References on the Runge-Kutta-Shanks Method

15. Shanks, E. B., "Formulas for Obtaining Solutions of Differential Equations by Evaluations of Functions," Presented at the summer meeting of the American Mathematical Society in Boulder, Colorado, August 1963, and Private Communication.

Other References

16. Ralston, A., "Runge-Kutta Methods with Minimum Error Bounds," Math. Comput., 16, (1962) 431-437.
17. Brush, D. G., Kohfeld, J. J., and Thompson, G. T., "Solution of Ordinary Differential Equations Using Two 'Off-Step' Points," J. ACM, 14, (1967) 84-89.
18. Scraton, R. E., "Estimation of the Truncation Error in Runge-Kutta and Allied Processes," The Computer Journal, 7 (1964) 246-248.
19. Butcher, J. C., "A Multistep Generalization of Runge-Kutta Methods With Four or Five Stages," J.ACM, 14 (1967) 84-89.
20. Kohfeld, J. J. and Thompson, G. T., "Multistep Methods With Modified Predictors and Correctors," J. ACM, 14 (1967) 155-166.
21. Gallaher, L. J., et al, "Study of the Methods for the Numerical Solution of Ordinary Differential Equations," Final Report Project A-831, Contract NAS8-20014, Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Georgia, (1967).